

AD-A188 826

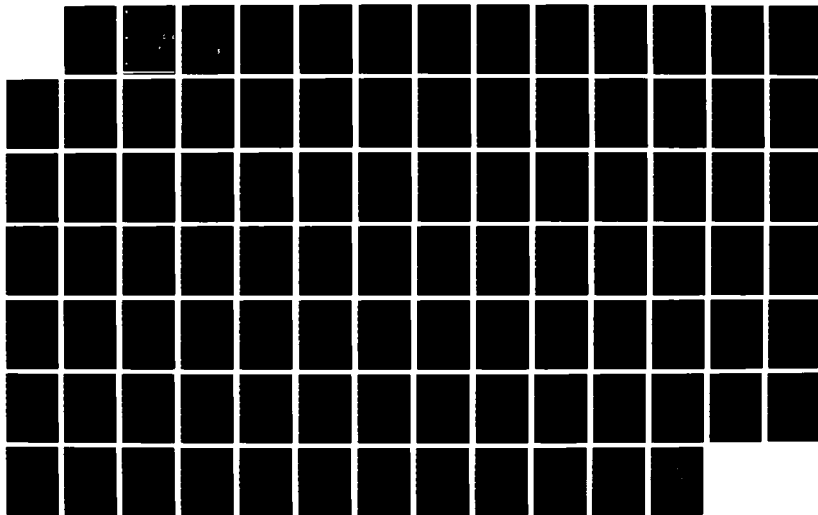
DESIGN METHODOLOGY FOR ALLOCATING DATA IN A DISTRIBUTED
DATABASE(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB
OH SCHOOL OF ENGINEERING E T POORE NOV 87
AFIT/GCS/ENG/87D-21

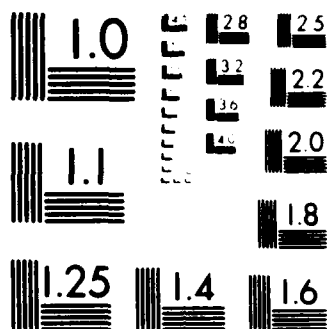
1/1

UNCLASSIFIED

F/G 12/7

ML





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A188 826



DTIC FILE COPY

DTIC
ELECTE
FEB 09 1988

Design Methodology for Allocating
Data in a Distributed Database

THESIS

Edward T. Poore Jr.
Captain, USA

AFIT/GCS/ENG/87D-21



DISTRIBUTION STATEMENT A

Approved for public release
Distribution Unlimited

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

88 2 4 068

AFIT/GCS/ENG/87D-21

Design Methodology for Allocating
Data in a Distributed Database

THESIS

Edward T. Poore Jr.
Captain, USA

AFIT/GCS/ENG/87D-21

DTIC
ELECT
FEB 09 1988
S D

Approved for public release; distribution unlimited

Design Methodology for Allocating Data in
a Distributed Database

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Computer Systems

Edward T. Poore Jr., B.S.
Captain, USA

November 1987

Accession For	
NTIS CRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Control
A-1	

Approved for public release; distribution unlimited



Preface

The purpose of this thesis was to develop a decision function to select the best method for allocating data in a distributed database, such as the United States Army Maneuver Control System. The reason for attempting this study was that Army management would benefit from a decision aid for choosing between various data allocation methods.

I have received much help in this effort. I wish to acknowledge Dr. Hartrum, my thesis advisor, who first enlisted my help and provided valuable guidance. I would also like to acknowledge my reader Major Daniel Reyen, for his help and instruction on linear programming techniques. Special thanks and a large measure of respect goes to my other reader, Captain Wade Shaw, who provided words of encouragement, greater insight into every aspect of my research, and an appreciation of what it means to be a true engineer. A special thanks to my friends, Fr. Joe, Debra, Patti, Cheryl, Beckie, Jeff and Beth whose faith in my abilities was a constant source of strength. Finally and most importantly, I wish to give thanks to my God and my family, whose love made getting through all the difficulties both possible and worthwhile.

Edward T. Poore Jr.

Table of Contents

	Page
Preface	ii
List of Figures	v
Abstract	vii
I. Introduction	1
Background	1
Definitions	2
Problem	4
Scope	4
Assumptions	5
Approach	7
II. Background	10
Summary of Current Knowledge	10
Chu's Allocation Approach	10
Morgan and Levin's Approach	12
Athans and Ma's Approach	13
Harwood's Approach	14
Conclusion of Analytical Allocation Approaches	15
III. Detailed Design Methodology	16
Introduction	16
Test Network Design Specifications	18
Data Allocation Methods	21
Transmission Cost Function Defined	21
Optimal Allocations	22
Heuristic Allocations	23
Automated Allocation Methods	24
Simulation of Test Network	24
Introduction	24
Model	25
Verification	27
Validation	27
Pilot Work	30
Output Analysis	33
Implementation Problems	36
Normalize Data for Decision Function	37
Execution of the Decision Function	38
IV. Results	40

Introduction	40
Output of Allocation Methods	40
Storage Cost	42
Resource Cost	42
Analysis of Simulation Data	43
Decision Function Normalized Data	54
Scenarios	55
Case A	57
Case B	57
Case C	58
Case D	58
Case E	59
Case F	59
 V. Conclusions and Recommendations	 62
Conclusions	62
Recommendations	63
Summary	64
 Appendix A: Source Code	 66
Appendix B: Detailed Analysis of Variance	67
Bibliography	81
Vita	82

List of Figures

Figure	Page
1.1 Network Configuration	6
3.1 Low Load Arrival Rates a) Update, b) Query	20
3.2 Network Resources	20
3.3 SLAM II Network Flow Diagram	26
3.4 Estimate vs. Model Output of System Criterion	30
3.5 SLAM II Sample Plot	31
3.6 High Load Arrival Rates a) Update, b) Query	34
4.1 Results of Allocation Methods a) Storage Allocation, b) Destination Node for Minimum Query Link	41
4.2 Complexity and Cost of Executing Allocation Methods	43
4.3 ANOVA Results a) R-Square Values, b) Full Model Factors, c) Simplified Model Factors	45
4.4 Duncan Grouping Main Effects	47
4.5 Duncan Grouping of Interactive Effects a) Low Load, b) High Load	49
4.6 Average System Times of Allocation Methods	50
4.7 Graph of Performance Criterion by Load a) Update Local, b) Update Remote, c) Query Local, d) Query Remote	52
4.8 Normalized Data for Decision Function a) System Performance Criterion, b) Storage and Resource Cost	55
4.9 Summary of Scenario's Weights	56
4.10 Output of Decision Function by Scenario	60
4.11 Ranking of Allocation Methods by Scenario	60
B.1 ANOVA Update Local for Null Hypothesis	68

B.2	ANOVA Update Remote for Null Hypothesis	69
B.3	ANOVA Query Local for Null Hypothesis	70
B.4	ANOVA Query Remote for Null Hypothesis	71
B.5	ANOVA Update Local for Significant Factors	72
B.6	ANOVA Update Remote for Significant Factors	73
B.7	ANOVA Query Local for Significant Factors	74
B.8	ANOVA Query Remote for Significant Factors	75
B.9	Duncan Range Test for Update Local Main Effect ...	77
B.10	Duncan Range Test for Interaction Update Local ...	77
B.11	Duncan Range Test for Update Remote Main Effect ..	78
B.12	Duncan Range Test for Interaction Update Remote ..	78
B.13	Duncan Range Test for Query Local Main Effect	79
B.14	Duncan Range Test for Interaction Query Local	79
B.15	Duncan Range Test for Query Remote Main Effect ...	80
B.16	Duncan Range Test for Interaction Query Remote ...	80

Abstract

The objective of this research was to find a method to solve the problem of deciding the best means for allocating data in a distributed data base. A decision function which considered system response times of update and query messages, storage cost of data items, and execution cost for allocating data was proposed. Additionally, the decision function allowed weighting of decision criteria to tailor the selection to specific network conditions. Five data allocation methods were evaluated for a given test network. The allocation methods were automated to facilitate future research efforts.

The test network was implemented as a simulation model consisting of six nodes and eight data items. The simulation model was both verified and validated. Statistical analysis was performed on selected outputs of the model. Normalized data from the simulation model, normalized output of the automated allocation methods and scenarios of weighted decision criteria for certain network conditions were evaluated by the decision function. The utility of the decision function was demonstrated through comparison of the results of each scenario. The impact of the research results were discussed and areas of future research were presented.

DESIGN METHODOLOGY FOR ALLOCATING DATA IN A DISTRIBUTED DATABASE

I. Introduction

Background

The Army Maneuver Control System (MCS) is a distributed command and control system, supporting the operations section (G3/S3) from the battalion to the Corps level. The planning cycle of the operations section is time critical. The MCS's response time to updates (changes in data) and queries (request for information) is essential to an efficient planning cycle.

The allocation of data in a distributed database affects the response time of updates and queries at each node in the distributed network. If data is locally replicated at several nodes, then a local query can be processed faster than a query requiring communication to another node for the required data. However, an update at one node must be

repeated for all other locations where the data is stored, requiring longer processing time than an update to a single storage location. An optimal allocation scheme for each data item in the database which minimizes use of the communications network subject to these conflicting time options is desired.

Often, algorithms which solve optimal allocation problems for large networks are cost prohibitive in terms of computation time or special resources required (such as a fast parallel processor). The designer of a network must still meet the user's response time needs for updates and queries to data items, while trying to maintain low utilization of the network which can change dynamically. Use of heuristics in assigning data items may reduce the cost of solving the problem, but the trade off may be other than an optimal solution. A thesis by Odus Harwood [Harwood] offered several possible heuristics for assigning data within the MCS.

Definitions

"A database management system (DBMS) consists of a collection of interrelated data or a set of programs to access that data. The key terms used to describe data bases are defined in the following discussion" [Korth and Silberschatz,1].

"The collection of data is usually referred to as the database" [Korth and Silberschatz, 1]. When the data is kept

at dispersed locations or several computers can interface with different parts of the data then this is a distributed database system [Ullman, 409].

A distributed database has several advantages. The overall storage requirement for the database is distributed over more than one node. Access time for data stored is reduced compared to access time over links to a centralized database node. For a given relation "r" there are different means to do the distribution.

Replication. The system maintains several identical replicas (copies) of the relation. Each replica is stored in a different site, resulting in data replication. The alternative to replication is to store only one copy of relation r.

Fragmentation. The relation is partitioned into several fragments. Each fragment is stored in a different site.

Replication and Fragmentation. This is a combination of the above two notions. The relation is partitioned into several fragments. The system maintains several identical replicas of each such fragment [Korth and Silberschatz, 408].

Replication of data at several nodes that need the data would decrease local processing time. However, updating the data is made more complex as it must be updated at all replicated sites. This increases the communication cost. Replication also increases reliability of the database system. "In some applications (e.g., military ones), even short down times due to hardware failures are completely intolerable, hence the need for redundancy" [Tannenbaum, 441].

In many distributed database management systems the multiple processors (computers) are connected by a communications network. The MCS is one such system. The connecting tactical communication system includes satellite channels, microwave links, and single channel FM radio. The speed of transfer of data from a disk or the computer's cache memory is much faster than the transfer of data over the tactical communications network. "The consequence of this assumption about communication is that the transfer of data between computers becomes a bottleneck, and most of the issues unique to distributed systems concern ways of dealing with this bottleneck" [Ullman, 409].

Problem

It is not known what the best method for allocation of data is for a distributed database such as the MCS. Given the nodes and their configuration in the network, the rate by node and data item of updates and queries, and the desired response time (performance criteria) for update and query transactions, the problem is to determine the methodology which most efficiently (in terms of computational time and cost of required resources) allocates data items within the performance criteria of the network and minimizes use of the communications network in terms of time.

Scope

This study uses a simulation model to evaluate three heuristic allocation schemes and two optimal allocation

schemes. The performance measures used in the comparison are average system completion time for an update and query message for both local and remote access to data.

The optimal allocation schemes for the two cases of nonredundant and redundant storage are based on a numerical algorithm solution. The heuristic methods include total replication, nonredundant storage based on the least update cost algorithm from Harwood's thesis, and nonredundant storage based on the greatest query rate algorithm, also from Harwood's thesis.

Assumptions

1. The Communications Network of Figure 1.1 is reliable.
2. There is sufficient storage available at each node to hold the required database. This removes storage as a constraint on the data allocation schemes.
3. The model uses estimated update and query rates and assumes Poisson distribution for arrivals, as actual rates and distributions of message arrivals for the MCS are not available.
4. The measurement criterion is based on average system values at steady state of an update or a query message which is processed locally or remotely.
5. The results obtained from this network can be generalized to similar configured networks with different parameters.

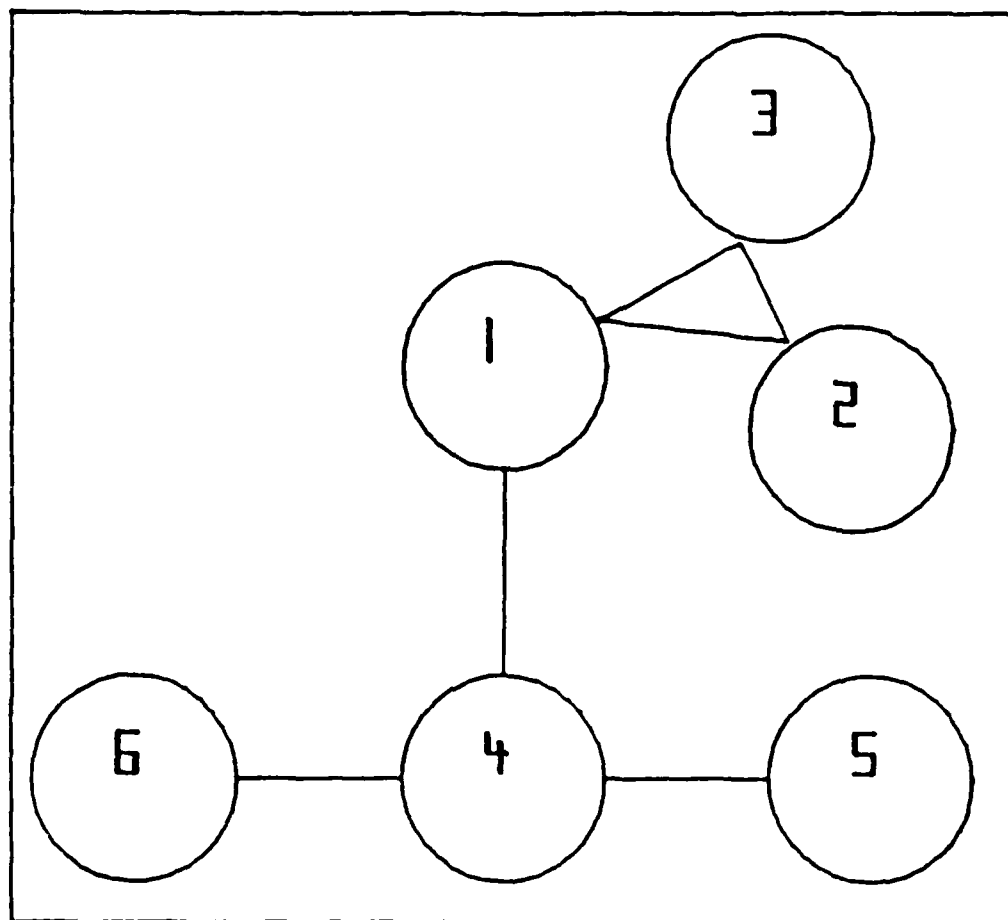


Figure 1.1. Network Configuration

Approach

A simulation model of a six node network was written in the simulation language SLAM II [Pritsker]. SLAM II was the language of choice due to its networking capability and its availability on several host computers. The model network parameters were held constant for each different allocation scheme, two system loads, and two queuing effects. This allowed comparison of chosen system parameters to the relative affects of each allocation method at two loads under two queuing disciplines.

The simulation model's output values for the selected system parameters were the basis of the allocation comparisons. Because of the importance of the model's output, the model had to be both verified and validated. Verification is ensuring that the code is functioning properly as intended. SLAM II trace and summary report tools were used to verify and debug network code.

Validation is checking that the right model was built. Comparison to actual system parameter measurements is the best method of validating a model's output. For this model it was not possible as actual data was not available. The focus of the thesis was on a design methodology rather than an exact model of the MCS. The comparison of relative goodness of the different allocation methods should still be valid for the given network, even if parameters turn out later not to match actual system times. The model's

parameters can be changed later to reflect more accurate system times as more data on the MCS becomes available.

After the initial model network was verified, the simulation was run with data based on the allocation methods holding the input load of update and query rates constant across each method and having the same queuing discipline for the central processing unit (CPU) resource at each node. The effect of a priority queue, based on message type, versus a first in first out (FIFO) queue for the CPU resource was also tested. The five allocation methods, two loading levels, and two queuing affects made for a total of 20 experimental conditions tested. The simulation was rerun based on each of the alternate conditions holding all other conditions constant.

The changes to the input rates for both updates and queries and location of each data item were coded in the initial model to be read from external ASCII files. The destination for routing and the source node for the minimum query path were fixed in SLAM II code requiring separate network code for each allocation method. The queuing discipline for the CPU resource was also hard coded in the SLAM II network code.

Pilot runs were made for each alternative to determine the length of each simulation run and a sufficient number of replication runs to have the data meet a statistical standard of 95% confidence and an error tolerance of plus or minus 15

seconds. The performance measures selected were compared for each of the different conditions, based on the simulation model output.

An analysis of variance of the simulation model's selected output tested the null hypothesis that there was no difference in the average system response times of update local, update remote, query local, or query remote response times due to the different allocation methods, loads, queuing disciplines, or their interactions. Duncan's range tests were performed on all main affects and any interactive affects where the null hypothesis was rejected. The range tests helped to gain insight into any statistical difference between allocation methods for a given set of conditions.

A decision model was proposed for selecting between the different allocation methods. Additional factors that were not directly considered in the cost function, such as storage cost of the allocation and resource cost to execute the allocation algorithms, were included in the decision model. The decision model was run for estimated weights or importance put on each factor in the decision model.

II. Background

Summary of Current Knowledge

Optimal allocation of data attempts to minimize the time required to process distributed database queries and updates by minimizing the cost of communication over the underlying network, subject to constraints put on the cost function of the network. If the information is not available at the local database, then a request must be made to the distributed database for the information. The required data must then be transferred over the communications network to the requesting node. If multiple copies of data are kept at nodes requesting it, the communication cost due to queries is reduced. The trade-off is the increased communication cost to maintain the concurrency of the redundant data. When the data item is changed, all duplicates of the item must be changed. The balance between these options in designing the distribution of data in a distributed database management system has been the object of much research.

Chu's Allocation Approach

Wesley W. Chu is one of the first to research file allocation with duplication in a model of a multiple computer system. According to Chu, "The overall operating cost related to the file is considered to consist of transmission and storage costs" [Chu, 885]. A set of equations for the cost between each node of the model represents the computer

system transmission cost and is the cost function which is to be minimized. This function is subject to constraints which involve the variables in the objective function. Chu constrained each node to hold only as many files as there was available memory. The optimal solution is the solution of these constraint equations which minimizes the overall operating cost equation.

Chu used the properties of his M/D/1 network to calculate the average queuing delay for each node. The M/D/1 notation means the message interarrivals are exponentially distributed, the queue server time is deterministic, and there is only one server. Each node's queuing delay due to message traffic was constrained to be less than a fixed upper bound. It is not possible to use Chu's problem formulation with networks which may have different distribution of server time or multiple servers. The calculation for more general network's queueing delay is not as straight forward nor always possible.

The transmission cost for requests in Chu's network depended on both where a request was originated and where the data items were stored. This resulted in "... solving a non-linear zero-one programming problem" [Chu, 887]. Chu transformed his cost function through additional constraints into a linear zero-one integer problem.

Zero-one or binary integer programming assigns a 1 to a variable if a condition is met (such as a file is allocated

to a node); otherwise it assigns a 0. Storage and transmission times are expressed in terms of the file allocation to a node. This type of problem is very complex and often is not solvable for an optimal solution. Chu added additional constraint equations to transform the problem into a linear zero-one programming problem, a problem solvable with known linear programming techniques such as branch and bound.

Although Chu gave a method to convert the problem to a linear zero-one programming problem, it required 2^N additional constraints where N is the number of nodes in the network. This makes converting large network problems impracticable, due the exponential growth of the required additional constraints.

Morgan and Levin's Approach

Howard Morgan and Dan Levin's model added complexity to the previous problem. Their model assumed a dependency between the location of the program requesting the data or initiating the update and the data's location in the model. The main costs considered were the communication cost of updates and queries and the storage cost. The resulting equation of their model was a non-linear zero-one programming problem [Levin and Morgan, 316-318]. "Applying a brute force method of reducing the non-linear problem to a linear zero-one problem would leave us with an unmanageable problem "[Levin and Morgan, 317]. The brute force method refers to

Chu's method of converting a non-linear to a linear zero-one program.

The individual allocation of each file was determined using a parallel processing machine, the hypercube [Levin and Morgan 317-319]. This approach did not seem promising for a real time application because a parallel processing architecture was needed to compute the optimal file allocation. The processing time to compute the file allocation on a sequential machine would be counterproductive to the actual required response times.

Athans and Ma's Approach

Michael Athans and Moses Ma attacked the problem of optimal file allocation when the computer network links are unreliable. The additional cost of not accessing the data is added to the storage cost and the communication costs. As in the case of Chu, their model resulted in an initial non-linear solution. Using additional constraint equations and a theorem developed in their paper, Athans and Ma reduced the problem to a linear zero-one program. For a reliable network an algorithm is given for optimal solution in polynomial time [Athans and Ma, 256-266].

The additional constraints required to reduce their problem to a linear zero-one program have the same limitations as for Chu's and Athans and Ma's problems. This limits this approach for use in very large networks.

Harwood's Approach

Harwood examined the allocation methods of the three approaches discussed thus far in this thesis. Harwood's conclusion was that "the models are difficult to understand and follow" [Harwood,43]. Harwood intuitively reasoned that heuristics were a better methodology for solving the MCS problem than an optimal approach. Harwood proposed six heuristic methods for allocating data in the network.

Harwood included in his cost function the separate update and query costs plus the storage cost. A minimum cost function was substituted for the communication link for a query from node j to node k [Harwood,120]. The algorithm for performing the minimum function was not given nor proposed.

Harwood's heuristics based on the update or query rates in the network are of a greedy type algorithm. The initial start point for each heuristic is to locate a data item at every node where the commander thinks it is critical. Provided that there is adequate storage at a location, a data item is added to a node which reduces either an update or a query cost without increasing the overall cost function. In the nonredundant version of his heuristics, the algorithm is terminated after one and only one noncritical data item above the commander's critical allocation is assigned to the network. In the redundant cases, data items are stored at noncritical nodes, as long as there is sufficient storage at the node and the overall cost function can be reduced.

Conclusion of Analytical Allocation Approaches

The reviewed analytic approaches to allocating data used linear extension programming tools to minimize a cost objective function. All models required difficult calculations without some additional constraints or simplification to the original model assumptions. Chu was able to reduce his model to a linear solution, but his model is limited in the number of nodes which can be evaluated without becoming unwieldy. Morgan and Levin's model requires exhaustive enumeration to arrive at an optimal solution. Athans and Ma's linear programming technique results in an algorithm which can be calculated within polynomial time for the reliable network case.

Harwood proposed several heuristics which are based on a cost function tailored to the MCS system. The goodness of these heuristics was not analyzed, so are good candidates for further research. An optimal solution for a small network should be calculated, so that relative goodness of the heuristic answers can be compared to the optimal answer.

III. Detailed Design Methodology

Introduction

There are many factors to be considered in selecting between data allocation methods. The load of the system, the average system times of the performance criterion, the data storage cost, and the cost to execute the allocation are all factors which affect the decision of choosing the best method among alternate allocation schemes. A low load on the system means no resource is utilized more than 25 percent. A high load represents the utilization of any one resource in the network approaching but not exceeding 95 percent. A decision function is needed that takes into account all of these factors and assigns a single relative weight to each allocation method to aid in a selection choice.

Such a proposed decision function is

$$\begin{aligned} \text{COST}[i] = & t_1 * (t_1 (h * \text{ALLOC}[2i+1,1] + 1 * \text{ALLOC}[2i+2,1]) \\ & + t_2 (h * \text{ALLOC}[2i+1,2] + 1 * \text{ALLOC}[2i+2,2]) \\ & + t_3 (h * \text{ALLOC}[2i+1,3] + 1 * \text{ALLOC}[2i+2,3]) \\ & + t_4 (h * \text{ALLOC}[2i+1,4] + 1 * \text{ALLOC}[2i+2,4])) \\ & + s * \text{ALLOC2}[i+1,1] \\ & + r * \text{ALLOC2}[i+1,2] \end{aligned} \quad (3.1)$$

where

$\text{COST}[i]$ is an array of results for each allocation method i tested.

ti is the weight placed on aggregate system response time.
 t1 is the weight placed on Update Local time.
 t2 is the weight placed on Update Remote time.
 t3 is the weight placed on Query Local time.
 t4 is the weight placed on Query Remote time.
 h is the weight placed on system at high load.
 ALLOC[] is an array holding data for system criterion. The first row of an allocation method is performance criterion data at low load and row two is data at high load.

Column 1 of each row is the average system time for Update Local. Column 2 holds Update Remote time. Column 3 holds Query Local time. Column 4 holds Query Remote time.

l is the weight placed on system at low load.
 s is the weight placed on storage cost of allocation method.
 ALLOC2[] is an array holding data for system storage cost and resource cost to allocate data in the system.
 r is the weight placed on resource cost in terms of computational complexity for execution of the allocation method.

Prior to executing the decision function to obtain relative costs of each allocation method, performance data and storage cost data must be collected or simulated for each allocation method being considered. A cost estimation for executing each allocation method must be obtained. A weight or importance of these normalized factors needs to be assigned by the manager or commander. Since actual data on the MCS was not

available, a simulated network was constructed to gather data running the simulation based on the data allocated by each method. This model data was used to test the null hypothesis stated in Chapter 1.

Test Network Design Specification

Several specifications were made to define the given test network model of Figure 1.1. The model represents a possible configuration of a Division's three tactical headquarters (nodes 1, 2 and 3) connected to one of the Division's Brigade tactical headquarters (nodes 4, 5 and 6). All other loads on this representation were ignored. The message rate into the model did not reflect any actual known input loads. The initial rates were based on reasonable levels. Combinations of all nodes having requests, partial number of nodes having requests, and changes in the ratio at a node of Update to Query messages were considered in the test message arrival rates for eight data items. The purpose of the combinations were to illustrate how and to gain insight as to why (based on arrival rates) the allocation schemes varied in assigning data to the network. All other numbers in the specifications had no basis from an actual network, but were merely reasonable estimates to form a base network to gather response time data for comparative analysis.

1. The message sizes are uniformly distributed between 250 and 2000 characters (2000 represents the possible number of ASCII characters on an 80 column 25 line monitor).

2. The operator typing time of original messages is 2.00 minutes per 2000 characters.
3. The CPU processing time of messages is Exponentially distributed with a mean of 0.0006667 minutes.
4. There are 12 average disk accesses per 2000 character message.
5. Disk access time is 0.0001667 minutes, which includes data transfer time.
6. The operator time to handle routing messages is 0.1 minute.
7. Each link in the network model is full duplex.
8. Message routing is fixed in the network.
9. The conversion factor for transmission time from a message size is 0.0001111 minute per character.
10. Each CPU burst is uniformly distributed.
11. The low load arrival rates of update messages by data item are in Figure 3.1. The rates are based on a given 24 hour period.
12. The low load arrival rates of query messages by data item are in Figure 3.1. These rates are also based on a given 24 hour period.
13. The CPU requires 0.001333 minutes to process the message routing routine.
14. The number of resources for CPUs, disks, and operators of each node is shown in Figure 3.2.
15. Duplicate update messages are created for each destination node and routed separately.
16. The cost of an Update, CU, from node j to node k is 1 times the number of links between the two nodes.
17. The cost of an Query, CQ, from node j to node k is 2 times the number of links between the two nodes.
18. The storage cost of placing a data item at a node is 1.

DATA	NODE						DATA	NODE					
	1	2	3	4	5	6		1	2	3	4	5	6
1	48	48	96	0	0	0	1	24	48	48	48	48	48
2	48	48	48	48	48	48	2	3	6	3	3	3	3
3	3	3	3	9	9	9	3	48	48	48	48	48	49
4	3	3	9	9	3	3	4	48	3	0	49	3	0
5	0	24	0	0	48	0	5	48	3	0	48	3	0
6	0	0	0	0	3	0	6	3	3	0	3	6	0
7	0	3	0	0	3	0	7	3	6	3	3	3	3
8	0	48	0	0	48	0	8	3	3	0	6	3	0

(a) Update

(b) Query

Figure 3.1. Low Load Arrival Rates a) Update, b) Query

NODE	OPERATOR	CPU/DISK
1	15	6
2	5	2
3	2	1
4	5	2
5	4	2
6	1	1

Figure 3.2. Network Resources

Prior to coding and testing the simulation network, the data allocation methods had to be implemented in order to have the data allocation input needed to run the simulation network. Where possible, the data allocation methods were automated for this network configuration.

Data Allocation Methods

The different allocation methods all considered network transmission costs and had the objective to:

1. Minimize the total network transmission time due to remote access of data.
2. Allocate each data item to at least one node in the network.
3. Determine storage requirements needed at each node.

Network communication costs include both the cost of update messages and the cost of query messages. The goal of all the allocation methods is to reduce the network transmission cost by reducing either the total query cost, the total update cost or both. The total transmission cost is defined in terms of the sum of the update and query costs for both the nonrecurrent and redundant cases.

Transmission Cost Function Defined

Let -

i = data item.

j = demand node (node where message request originates).

k = source node (node where data item is stored).

CU_{jk} = Cost of Update from node j to node k .

CQ_{jk} = Cost of Query from node j to node k .

RU_{ij} = Rate of Updates for data item i generated at node j .

RQ_{ij} = Rate of Queries for data item i generated at node j .

X_{ik} = binary decision variable for storage of data item i at node k . 0 is assigned if data item not stored and 1 is assigned if data item is stored.

The network Update cost is

$$\sum_i \sum_j \sum_k CU_{jk} * RU_{ij} * X_{ik} \quad (3.2)$$

The network Query cost (nonredundant) case is

$$\sum_i \sum_j \sum_k CQ_{jk} * RQ_{ij} * X_{ik} \quad (3.3)$$

The network Query cost (redundant) case is

$$\sum_i \sum_j \sum_k \min (CQ_{jk} * RQ_{ij} * X_{ik}) \text{ for } j \text{ not equal } k \quad (3.4)$$

The total network cost (nonredundant case) from equations 3.2 and 3.3 is

$$\sum_i \sum_j \sum_k ((CQ_{jk} * RQ_{ij} + CU_{jk} * RU_{ij}) * X_{ik}) \quad (3.5)$$

The total network cost (redundant case) from equations 3.2 and 3.4 is

$$\sum_i \sum_j \sum_k (CU_{jk} * RU_{ij} * X_{ik}) + \sum_i \sum_j \sum_k \min (CQ_{jk} * RQ_{ij} * X_{ik}) \quad (3.6)$$

Optimal Allocations

For the optimal nonredundant case equation 3.5 was minimized subject to the constraints that X_i is a binary integer and that $\sum_k X_{ik} = 1$ for each data item i . This means each data item i has to be stored once and only once in the network. The trivial and impractical solution to not store any

data items, making the transmission cost zero, was eliminated by these constraints.

For the optimal redundant case equation 3.6 was minimized subject to the same binary constraint as in the nonredundant case. The assignment constraint, $\sum_k X_{ik} = 1$, was relaxed to allow for a sum greater than one for each data item i . This means each data item i has to be stored at least once, but may be stored more than once. For this test network, there were 63 combinations of storing a data item in a six node network at least once.

Heuristic Allocations

Total redundant allocation was a trivial case in the sense that it reduced the total transmission cost to just the update cost equation 3.2. Since every demand node j was also the source node, the query cost was zero. However, this heuristic also had the distinction of maximizing the network update cost.

The Least Update Rate nonredundant allocation "seeks to allocate data items to those nodes where the sum of the update rates to that data item from all other nodes is the least" [Harwood,137]. Equation 3.2 was adjusted to find the source node k which minimized $(\sum_j C_{Ujk} * R_{Uij} * X_{ik})$, where k is not equal to j . The data item was stored at this node.

The Greatest Query Rate nonredundant allocation attempts to reduce the network transmission cost for each data item by storing the data item at the demand node with the greatest request rate. This would eliminate any query transmission cost

at the highest demand node for that data item. This required finding the max value of the RQ_{ij} term of equation 3.4 for each data item i and storing the data item at the demand node j for that value.

Automated Allocation Methods

The cost functions that applied to each allocation scheme were coded in the C program language. Each automated allocation method read the network cost and matrices for Update and Query messages from external ASCII files. The optimal redundant allocation also read the 63 possible storage combinations from an external file.

Each automated method output consisted of two ASCII files. One file contained the information on the binary decision variable X_{ik} for each data item by node. The storage requirement for each allocation method was obtained by summing all the binary decision variables in this file. The other file contained the information of what source node would give the minimum query cost for a given data item by demand node. These two files were used as input to the simulation of the test network.

Simulation of Test Network

Introduction

A simulation model of the test network was designed. The model input data was generated. The model was first verified and then validated. Upon validation, pilot work was performed to determine the number and lengths of the simulation runs.

This pilot work was done for each combination of parameters that was tested. Given the correct run length and number of simulations to run, the output data of the simulation model was collected and analyzed.

Model

Figure 3.3 shows a flow diagram of the SLAM II model of the test network. Input messages were created for both Update and Query messages for each node. For this network of 6 nodes and 8 data items there were 48 update and 48 Query message generators. After a message was created parameters were assigned, such as source node, destination node, message length and message start time. After the parameters were assigned, the message was sent to the node logic for further processing and routing.

In the node logic, the operator resource typed all original messages. If a message was not designated for that node, it was routed through the network to the correct designation node. If a message destination was the same as that node, the message was processed and the system time was collected. Message processing time included the CPU processing at that node and any required disk access time to obtain or change information stored at the node.

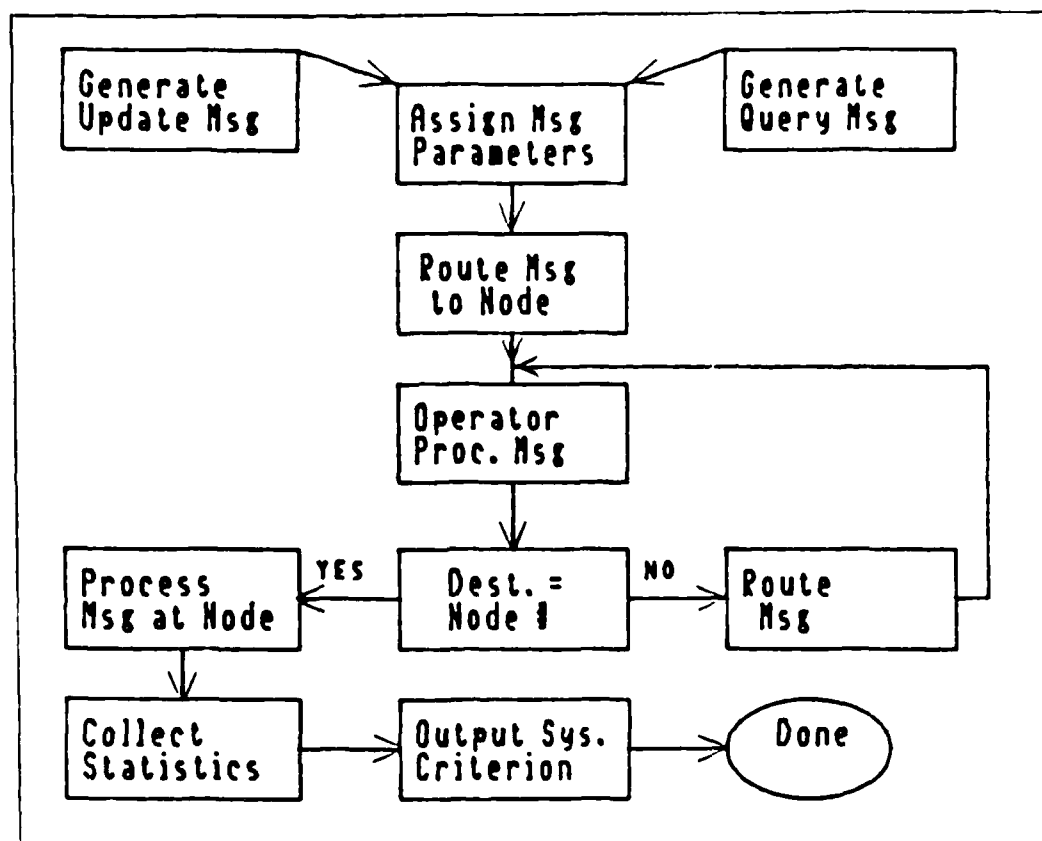


Figure 3.3. SLAM II Network Flow Diagram

The simulation used three external ASCII files for each allocation method. The arrival rates for Update and Query messages were read from two external files and converted to interarrival times used by the message generators. Storage location of data items in the network was read from an external file and was used to help determine if a message was a local or remote message. The destination node for the minimum

communications cost for a given demand node remote message was coded internally in the control section the SLAM Network. This required separate control sections (the network logic code remained the same) for each allocation method.

Verification

Verification of the SLAM model was obtained through functions and reports which were a part of the SLAM language. A monitoring function in SLAM allowed a message to be traced and changes in its parameters to be displayed. Parameter assignments and correct routing of messages through the model code were verified with this monitoring function. SLAM also provided a summary report which gave statistics on resource utilization. Verification of the utilization of all resources needed to process a message or route a message was obtained with the summary report.

Validation

The validation step of simulation was difficult due to the lack of actual system values or a tractable mathematical model output to compare values generated by the model. Instead, reasonable estimates were made for expected values for Update and Query messages processed both locally and remotely. The system times were estimated using the test network specifications and a few simplifying assumptions. These estimates were then compared to the output system criterion of the model for the optimal nonredundant case at low system load.

The estimates were all based on average values. The average length of a message, from network specification 1, is 1125 characters. The average time to type a message is 1.125 minutes, from specification 2 and the average message length. The average number of disk accesses is 6.75 with an average disk access time of .0011 minutes, from specifications 4 and 5 and the average message length. The average CPU message process time is .000667 minutes, from specification 3. The average one-way routing time for one link is 0.2263 minutes (including transmission time - 0.125 minutes, operator routing time 0.1 minute, and CPU routing overhead - 0.0013 minutes), from specifications 6, 8, and 12 and the average message length.

The queuing effect of the network and the average number of links used in processing a remote message were two unknown factors needed for completing the estimates. It was assumed at low system load that the queuing influence would be minimal. It was also assumed that, on the average, only one and a half links from a demand node to a source node were needed to process remote message traffic.

For the local messages (both Update and Query) the estimated time included the typing time, CPU message processing time and disk access time. The estimated local system time was 1.126 minutes. The remote Update message included the local processing time plus the additional time of one transmission. The Update remote message estimated average system time was 1.465 minutes. The remote Query estimated system time had an

additional transmission cost for the return of the request information. Its estimated average system time was 1.805 minutes.

These estimate times were compared to the output system times of the simulation for the optimal nonredundant allocation at low load. Figure 3.4 shows a summary of the comparison data. It was expected that the variance between the estimates and the model's output for locally processed messages would be low. The model's output average for Update and Query messages processed locally were expected to be relatively the same. The model average for a Query remote message was expected to be higher than the Update remote message and both of these averages were expected to be higher than the local message averages. It was also expected that the variance between the model output and the estimate for an Update remote message would be higher than the local messages and lower than the variance for the Query remote message. Figure 3.4 clearly shows the expected results and trends, thus validating the model. The model output criteria for the Update local message was 1% higher than the estimate. The output for the Query local was 2% higher than the estimate. The difference between the model's averages for Update and Query messages was less than 1%. The output for the Update remote was 4% higher than the estimate. The output for the Query remote was 8% higher

than the estimate. The variance of the model's output from the estimates was attributed primarily to random error and the minimal queuing effect of the network.

	Update Local	Update Remote	Query Local	Query Remote
Estimate	1.126	1.465	1.126	1.805
Model Output	1.139	1.529	1.146	1.954

Figure 3.4. Estimate vs. Model Output of System Criterion

Pilot Work

The collection of statistics, which were the performance criteria of average system time to process Update and Query messages both locally and remotely, required steady state conditions. The length of the simulation run had to be long enough to ensure this condition was met. SLAM's plot function was used along with a user written function to display the average value of each criterion time over time. When the curve of the plot for each average system time flattened out, the steady state value was reached. Figure 3.5 shows a sample plot from a pilot run for the total redundant allocation method. The length of the simulation could be set using this plot.

In Figure 3.5 all three curves were flattened out at 720 minutes into the simulation. The steady state values for these three statistics were available any time after this. The plot for Query remote average system time is not shown, because this allocation had no remote queries. Similar plots were

required for each set of conditions tested by a given simulation run to determine that simulation's run length.

		SCALES OF PLOT						
U=UPDATE		LOCAL	0.000e+00				0.150e+01	
Q=QUERY		LOCAL	0.000e+00				0.150e+01	
R=UPDATE		REMOTE	0.000e+00				0.150e+01	
		0	10	20	30	40	50	DUPS
MINUTES								
0.0000e+00	U							+ UQ UR
0.6000e+02	+				Q	U	R	+
0.1200e+03	+				QU	R		+
0.1800e+03	+				QU	R		+
0.2400e+03	+				U	R		+ UQ
0.3000e+03	+				QU	R		+
0.3600e+03	+				QU	R		+
0.4200e+03	+				QU	R		+
0.4800e+03	+				QUR			+
0.5400e+03	+				QUR			+
0.6000e+03	+				QUR			+
0.6600e+03	+				QUR			+
0.7200e+03	+				UR			+ UQ
0.7800e+03	+				UR			+ UQ
0.8400e+03	+				UR			+ UQ
0.9000e+03	+				UR			+ UQ
0.9600e+03	+				UR			+ UQ
0.1020e+04	+				UR			+ UQ
0.1080e+04	+				UR			+ UQ
0.1140e+04	+				UR			+ UQ
		0	10	20	30	40	50	DUPS
MINUTES								

Figure 3.5. SLAM II Sample Plot

Pilot work was also needed to determine the number of simulation runs. Many factors affected the number of runs made such as the time it takes to complete multiple runs. Most

importantly, the acceptable confidence interval in using the observed model system times and the statistics generated from those samples was the major concern.

For this study a 95% confidence interval was the acceptable level. This means if 100 experiments were conducted, 95 would generate a confidence interval which included the true population mean. The difference between the sample mean and the true expected value is the error. The larger the sample size (in other words the more runs) the smaller the error distance. The given acceptable error for this study was 0.25 minutes.

Appealing to the Central Limit Theorem, the sampling mean of a large number of independent observations (random variables) should approximate a normal distribution. A large number is usually taken to be 30 or more. For smaller numbers of random variables the t distribution can be used. Knowing the level of confidence and the acceptable error the following formula was used to estimate the number of runs :

$$\text{runs} = ((t * s) \setminus e)^2 \quad (3.7)$$

where t is the t distribution value for $\alpha \setminus 2$
 and n-1 degrees of freedom

 s is the standard deviation of the sample
 population.

 e is the acceptable error.

The largest number of runs needed for any one of the combinations of parameters tested was the number of runs used

by all of the runs. For this study, 6 runs met the given requirements.

Pilot work was also used along with the SLAM summary report to determine what was a high load for this network. Plots of system times for loads above 5 times the low rates used did not reach steady state condition even after 5760 minutes of simulation time. The summary report, for this system load, showed high utilization of the operator resources at node 6 and the communication link from node 1 to node 4, which indicated steady state condition would not be reached for all allocation methods. At four times the low rates, steady state was achieved, but required run lengths of 2160 minutes. The high load for this system was set at four times the low rates of Update and Query messages. The high rates are shown in Figure 3.6 for Update and Query.

Output Analysis

There were a total of 20 different combinations of 5 allocation methods, 2 system loads, and two queuing disciplines tested. With the number of runs for each combination being 6, the total number of data points for each system criteria was 120. All the data points for the 4 system performance measures were combined into one file for analysis.

DATA	NODE					
	1	2	3	4	5	6
1	192	192	384	0	0	0
2	192	192	192	192	192	192
3	12	12	12	36	36	36
4	12	12	36	36	12	12
5	0	96	0	0	192	0
6	0	0	0	0	12	0
7	0	12	0	0	12	0
8	0	192	0	0	192	0

(a) Update

DATA	NODE					
	1	2	3	4	5	6
1	96	192	192	192	192	192
2	12	24	12	12	12	12
3	192	192	192	192	192	196
4	12	12	24	12	12	12
5	192	12	0	196	12	0
6	192	12	0	192	12	0
7	12	12	0	12	24	0
8	12	12	0	24	12	0

(b) Query

Figure 3.6. High Load Arrival Rates a) Update, b) Query

Analysis of variance (ANOVA) was the technique used to test the null hypothesis that there was no effect due to the three factors of allocation methods, system load, queuing discipline, or their interactions on the system performance criterion. Statistical Analysis System (SAS), a computer program developed by the SAS Institute of Cary, North Carolina [Cody and Smith], was used to execute the ANOVA for this three factor experiment. Where the null hypothesis was rejected, multiple-range tests were performed to indicate which means differed significantly.

Multiple-range tests essentially test all hypotheses that there is no difference between two or more means for a given controlled significance level, [Walpole and Myers, 365]. Duncan's multiple-range test, available as a procedure in SAS, provided the analysis of which means, for a given set of factors, differed significantly. Duncan's test assumes " k random samples are all of equal size n . The range of any subset p sample means must exceed a certain value before we consider any of the p population means to be different" [Walpole and Myers, 365]. Tables and a formula to help manually calculate the critical values can be found in [Walpole and Myers, 365, 474]. SAS calculates the critical values for the Duncan multiple-range test internally within its program procedure.

Implementation Problems

There were several inconvenient implementation problems. The combined Fortran and SLAM II code had to be moved in order to overcome some computer filing system and execution time problems which occurred while using two of the Air Force Institute of Technology's (AFIT) Unix based computers. Different Fortran 77 compilers used on the Unix based computers required some code modification for portability between machines. Lack of a good statistical computer package on the Interim Computer Capacity (ICC), an Elxsi 6400 computer available at AFIT, required transferring files to other computers at AFIT which had statistical packages.

The first problem was to move the combined Fortran and SLAM II code of the simulation from the Academic Support Computer and the Scientific Support Computer (both VAX 11/785 computers running Unix operating system) to the ICC. The ICC was chosen as the target machine, because the other Unix based machines were having problems with the filing systems being over capacity and the combined Fortran and SLAM II code executed faster on the ICC. Three sets of simulation runs were completed in one day on the ICC whereas on the other Unix based machine, not even one of the sets was finished by the end of the duty day.

The second problem was with the Fortran 77 compiler used on the ICC. This version when opening a file required an

additional rewind statement before the file could be read. The Fortran code from the other Unix based machines was modified to run on the ICC.

The third problem was the execution of SLAM's CREATE node [Pritsker, 112-115] used to generate the simulation network's message traffic. A beginning value to start the message generation had to be declared and a variable is not allowed for this beginning time. Zero was the chosen start value and at this time every message generator created a message. This essentially was flooding the network at time 0. A special code was added to terminate all messages with a start time less than 1 minute. This sifting of the effective start of the simulation did not affect any of the output values and corrected the simulation start up problem.

The fourth problem was the lack of any good statistical computer program on the ICC. This problem was a minor inconvenience as the file transfer at AFIT was used to transfer data from the simulation to the Classroom Support Computer (CSC), a VAX 11/785 using the VMS operating system. The CSC has the SAS package used in the output analysis of the simulation.

Normalize Data for Decision Function

The multiple costs associated with system performance, storage, and resource did not all have the same dimensions. System performance criterion was in minutes and resource cost was in number of operations. These summed factors made no

sense with these dimensions. There was also a problem of scale with the different decision criteria. Storage costs ranged from 8 to 48 units, while resource costs ranged from 288 to 147456 operations. The mere size of the resource cost would have given it more weight than the decision criteria of storage cost. Normalizing the data on a scale from 0 to 1 made the decision function dimensionless and eliminated the problem of scaling between different costs or performance criterion.

The data was normalized by dividing each decision criteria data set by the highest value in the given data set. The averages of every allocation method for an Update local message at high load were divided by the highest average of those five averages. The storage costs for each allocation method were divided by the highest storage cost of 48 units.

Execution of the Decision Function

Once the data was normalized for the decision function, equation 3.1, the weights needed to be applied before executing the decision function. To insure the output was on a scale from 0 to 1, constraints were placed on values assigned to the weighting factors of the decision function. The first constraint was the weights of importance on the aggregate system time, storage cost, and resource cost had to sum to 1. The second constraint was that the weights applied to the system load had to sum to 1. The last constraint was that the weights applied to the system performance criterion of Update local, Update remote, Query local, or Query remote all

had to add to one. The second and third constraint only affected the overall value of the aggregate system performance time.

For a given set of weights the decision function was executed. The output of the function for each allocation method was analyzed to determine a selection. The lowest value of the decision function for an allocation method was the number 1 choice among the different allocation methods for those weightings.

IV. Results

Introduction

There were many intermediate results leading up to the execution of the final decision model. The results of the allocation cost function were used as input to the simulation model. The results of the simulation model runs were averaged and normalized for use as input parameters to the decision model. The storage data output of the allocation methods were totaled for each method to determine its overall storage cost. Harwood's computational complexity analysis of different allocation methods [Harwood, 205-209] was used to determine the resource cost (in terms of number of operations) to execute each allocation method. Both the storage and resource costs were normalized and these data were used as parameters in the decision model. Several scenarios of a network and the weights placed on the parameters were considered. The decision model was executed for each scenario and the results were tabulated.

Output of Allocation Methods

The automated allocation methods indicated storage of a data item at a node by placing a 1 in the storage file for that node or else a 0 was stored in the file. The automated methods also indicated for each demand node for each data item what source node would provide a minimum query transmission cost. Figure 4.1 gives a summary of storage allocation and

destination nodes for all 5 allocation methods. The symbol for the allocation method represents the 1 in the storage file.

	NODE					
DATA	1	2	3	4	5	6
1	*C	* Q	**L	* +	*	*
2	*C+L	* Q	*	*	*	*
3	* +	**	**	*C+L	**	**Q
4	*C+L	*	* Q	*	*	*
5	*	*	*	*C+ Q	* L	*
6	* + Q	*	*	*C+L	**	*
7	* L	**	*	*C	** Q	*
8	* L	*	*	*C+ Q	*	*

(a) Storage Allocation

	NODE					
DATA	1	2	3	4	5	6
1	11332	21332	31332	41432	51432	61432
2	11112	21112	31112	41112	51112	61112
3	14146	24246	34346	44446	54546	64646
4	11113	21113	31113	41113	51113	61113
5	14454	24454	34454	44454	54454	64454
6	14141	24141	34141	44441	54541	64441
7	14215	24215	34215	44515	54515	64515
8	14414	24414	34414	44414	54414	64414

(b) Destination Node for Minimum Query Link

LEGEND:

CODE	ALLOCATION METHOD
*	TOTAL REPLICATION
C	OPTIMAL NONREDUNDANT
+	OPTIMAL REDUNDANT
L	LEAST UPDATE COST
Q	GREATEST QUERY RATE

Figure 4.1. Results of Allocation Methods a) Storage Allocation, b) Destination Node for Minimum Query Link

Figure 4.1a shows that for the total replication allocation method (symbol "**") data items 1-8 were stored at every node. For the optimal redundant allocation method Figure

4.1a shows that data item 1 was stored at nodes 3 and 4. It is clear that the various allocation methods did not store all data items at the same nodes.

Figure 4.1b indicates that query requests for data item 1 at node 1 for the optimal redundant method had node 3 as the source destination. Query requests at nodes where the source node was not the same as the demand nodes were remote queries. Figure 4.1b showed that for the total replication allocation method the source node was the same as the demand node request in all cases. There were no remote queries for this allocation method.

Storage Cost

The storage cost for each allocation method can be easily obtained by summing the number of each type symbol in Figure 4.1a for each allocation method. Total replication was the most expensive allocation in terms of storage; 8 data items were each stored at six nodes for a total cost of 48. The next allocation method most expensive, in terms of storage, was the optimal redundant with a cost of 17. All other allocations being nonredundant stored each data item only once in the network for a storage cost to each of 8.

Resource Cost

Harwood's computational complexity analysis of different allocation methods [Harwood, 205-209] was used to determine the resource cost (in terms of number of operations) to execute each allocation method. The only method not analyzed by

Harwood was for the optimal nonredundant case. This case from equation 3.5 includes both Update and Query costs of the network. These two costs together are a form of Harwood's total cost function. "It was determined previously that $(M^2N + MN)$ operations were required for computing total costs" [Harwood, 205]. Figure 4.2 is summary of Harwood's analysis and the big "O" cost for a 6 node network with 8 data items.

Allocation	Big "O"	Cost
Total Replication	$O(M^2N)$	= 288
Optimal Nonredundant	$O(M^2N + MN)$	= 336
Optimal Redundant	$O(2^M(MN)^2)$	= 147456
Greatest Query Rate	$O((MN)^2)$	= 2304
Least Update Cost	$O((MN)^2)$	= 2304

Legend:

M is # of Nodes (6)

N is # of Data items (8)

Figure 4.2 Complexity and Cost of Executing Allocation Methods
(Source: [Harwood, 205-209])

Analysis of Simulation Data

All the output data from the simulation model was placed in one data file. SAS, a statistical package, was used to perform an analysis of variance (ANOVA) which tested the null hypothesis. There were several values given in the ANOVA tables. One value which indicated how well the model described the variance of the data was the R-Square value. The R-Square values were close to the value 1 and thereby reflected good

models. The other important value was the probability associated with the F value calculated for each factor of the model. This probability was expressed as either significant at the test level of 5% or else not significant.

The first ANOVA table with all factors and their interactions showed that the queuing factor was not significant nor were any interactions with the queuing factor significant at the 5% level. The model was changed to confound the queuing factor and all its interactions with the error sum of squares. The simpler model was used to produce a new ANOVA table. Figure 4.3 is a summary of the two important values from the ANOVA tables for both the full and simplified model. The detailed SAS ANOVA tables and associated Duncan multiple-range tests can be found in Appendix B.

Figure 4.3a showed that, because the simpler model's R-square values are essentially the same as the full models and the R-square values were high, the simpler model could be used to explain the variance of the data. Figure 4.3b further showed that for all system criterion times, the queuing factor and its interactions were not significant. This meant there was insufficient evidence to reject the null hypothesis on the effect of queuing and its interaction.

VARIABLE	FULL MODEL	SIMPLIFIED MODEL
T1	0.867	0.863
T2	0.950	0.950
T3	0.840	0.838
T4	0.992	0.992

(a) R-Square Values

SOURCE	DF	T1	T2	T3	T4
ALLOC	4	*	*	*	*
LOAD	1	*	*	*	*
ALLOC*LOAD	4	*	*	*	*
QUEUE	1	-	-	-	-
ALLOC*QUEUE	4	-	-	-	-
LOAD*QUEUE	1	-	-	-	-
ALLOC*LOAD*QUEUE	4	-	-	-	-

(b) Full Model Factors

SOURCE	DF	T1	T2	T3	T4
ALLOC	4	*	*	*	*
LOAD	1	*	*	*	*
ALLOC*LOAD	4	*	*	*	*

(c) Simplified Model Factors

LEGEND:

ALLOC ALLOCATION
 T1 UPDATE LOCAL
 T2 UPDATE REMOTE
 T3 QUERY LOCAL
 T4 QUERY REMOTE
 * SIGNIFICANT AT ALPHA LEVEL OF 5%
 - NOT SIGNIFICANT

Figure 4.3. ANOVA Results a) R-Square Values, b) Full Model Factors, c) Simplified Model Factors

Figure 4.3b and Figure 4.3c both showed that all other factors and their interactions were found to be significant. There was sufficient evidence to reject the rest of the null hypothesis and accept the alternate hypothesis. There was a difference between the allocation methods. There was a difference between high and low load. And there was a difference based on the interaction of load and allocation method. Because the null hypothesis was rejected, a post hoc test was needed to test which factors were different from each other.

The Duncan multiple-range test is a post hoc test which compares the difference between two or more means. If the difference is less than a least critical value than there is no significant difference among those group of means. Lines on the same level in the main factors Duncan grouping of Figure 4.4 were not significantly different. The highest level line is the worst mean time and the lowest level line is the best mean time. The best mean time in this study was the smallest mean time. For Update local message the total redundant allocation was worse than all the other allocations. The optimal redundant was second worst for the same criterion. There was no significant difference between the mean Update local message processing time for the greatest query rate allocation and the least update cost allocation, but both allocations were significantly worse than the optimal nonredundant allocation method which had the best mean time.

Duncan Grouping							
Criterion	Allocation					Load	
	A	M	Q	L	S	High	Low
Update Local	—	—	—	—	—	—	—
Update Remote	—	—	—	—	—	—	—
Query Local	—	—	—	—	—	—	—
Query Remote	—	—	—	—	—	—	—

Legend:
A TOTAL REDUNDANT
M OPTIMAL REDUNDANT
Q GREATEST QUERY RATE
L LEAST UPDATE COST
S OPTIMAL NON-REDUNDANT

Figure 4.4. Duncan Grouping Main Effects

From Figure 4.4 it was clear that different allocation methods were better than others depending on what system criterion was considered. For the Query remote overall average the total redundant allocation gave the lowest time with either of the optimal allocations having the second lowest time. However, for Query local the two optimal allocations had the best average time and the total redundant allocation had the worst average time of all the allocations.

As in Figure 4.4, the lines on the same level in the interactive factors Duncan grouping of Figure 4.5 were not significantly different. The interactive factors were allocation and load. Because of the significance of the interaction, the interpretation of the Duncan grouping for the main effects do not hold in all cases. There were allocation methods that had better or no significant difference of system response times at high load versus low load. The Duncan grouping main effects showed that there was a significant difference between high and low load with the high load on the average having the worst response time.

The significance of the interaction between load and allocation made the analysis more interesting, if not more difficult. The interaction meant the grouping of at least one of the allocations would change depending on the load or else the grouping of the loads would change depending on at least one of the allocations. The clearest example of this in Figure 4.5 was for the Update Local criteria. At low loads there was no significant difference between which allocation was used. However, at high loads the optimal nonredundant allocation was definitely better than the total redundant allocation.

Duncan Grouping					
Allocation @ Low Load					
Criterion	A	M	Q	L	S
Update Local	_____				
Update Remote	_____	_____	_____	_____	_____
Query Local	_____				
Query Remote	_____	_____	_____	_____	_____

Legend:

A TOTAL REDUNDANT
M OPTIMAL REDUNDANT
Q GREATEST QUERY RATE
L LEAST UPDATE COST
S OPTIMAL NON-REDUNDANT

(a) Low Load

Duncan Grouping					
Allocation @ High Load					
Criterion	A	M	Q	L	S
Update Local	_____	_____	_____	_____	_____
Update Remote	_____	_____	_____	_____	_____
Query Local	_____	_____	_____	_____	_____
Query Remote	_____	_____	_____	_____	_____

Legend:

A TOTAL REDUNDANT
M OPTIMAL REDUNDANT
Q GREATEST QUERY RATE
L LEAST UPDATE COST
S OPTIMAL NON-REDUNDANT

(b) High load

Figure 4.5 Duncan Grouping of Interactive Effects a) Low Load, b) High load

The interaction effect between the load and the allocation was not always obvious from Figure 4.5. An example was for the Query remote time. The grouping of allocations was the same for both high and low loads. The interaction was with the load and the total replication allocation method. The main effect of load indicated that high load performance was significantly worse than low load. However, for the total replication method there was no significant difference for Query remote processing time based on load, as the time for both loads was 0. The actual values of the means in the Duncan multiple-range grouping shown in Figure 4.6 were needed to see this interaction.

DECISION		ALLOCATION METHOD				
PARAMETER	\	A	S	M	L	Q
T1 LOW LOAD		1.17	1.14	1.15	1.15	1.15
T1 HIGH LOAD		1.49	1.12	1.30	1.20	1.24
T2 LOW LOAD		1.13	1.53	1.26	1.56	1.63
T2 HIGH LOAD		2.19	1.66	1.49	1.73	1.82
T3 LOW LOAD		1.15	1.15	1.15	1.15	1.16
T3 HIGH LOAD		1.49	1.13	1.24	1.19	1.38
T4 LOW LOAD		0.00	1.95	2.00	2.11	2.35
T4 HIGH LOAD		0.00	2.33	2.38	2.54	3.06

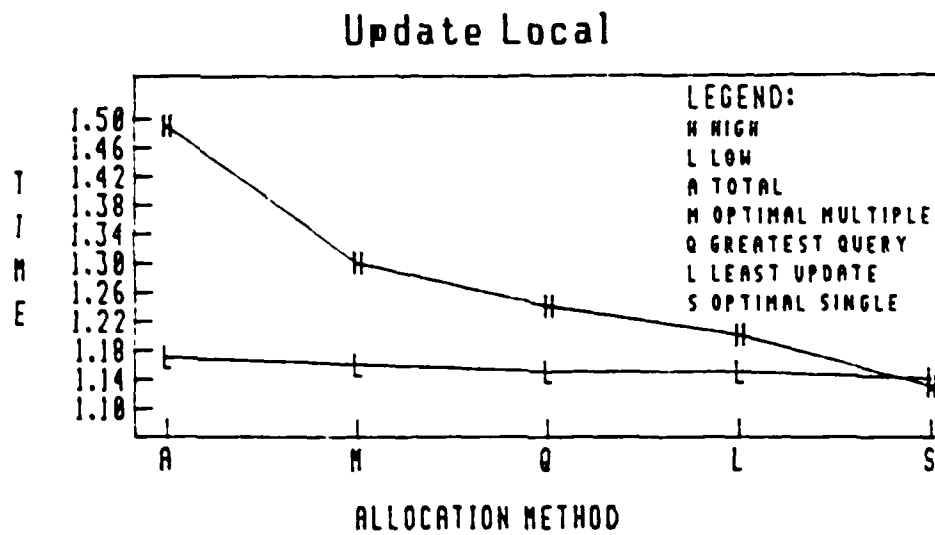
LEGEND:			
CODE	PARAMETER	CODE	ALLOCATION
T1	UPDATE LOCAL TIME	A	TOTAL REPLICATION
T2	UPDATE REMOTE TIME	S	OPTIMAL NONREDUNDANT
T3	QUERY LOCAL TIME	M	OPTIMAL REDUNDANT
T4	QUERY REMOTE TIME	L	LEAST UPDATE COST
		Q	GREATEST QUERY RATE

Figure 4.6. Average System Times of Allocation Methods

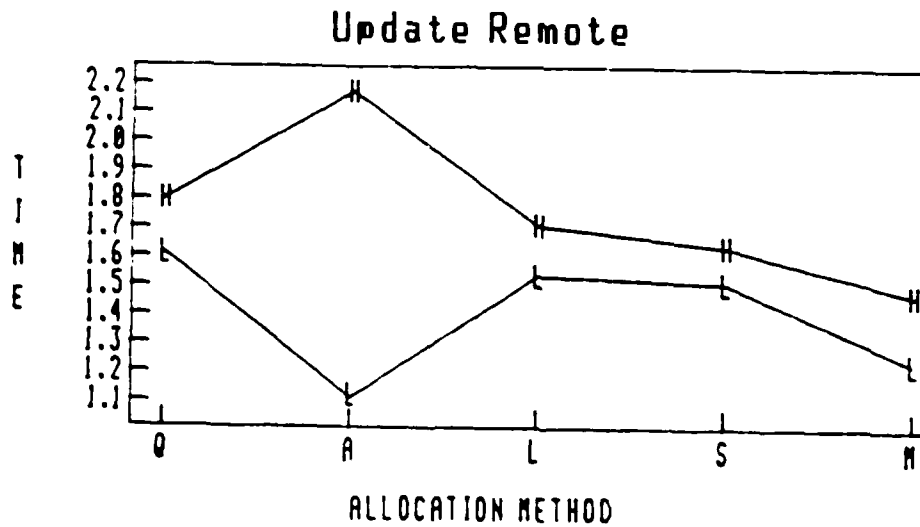
The data in Figure 4.6 was more informative when displayed graphically. Figure 4.7 shows the graphical representation of each performance criterion by load. Some of the interactions are more obvious with the graphs of Figure 4.7 than with the Duncan groupings of Figure 4.5. The allocation methods on the x axis of the graph were listed by their main factor Duncan grouping from worst to best, with the worst allocation time being closest to the y axis.

Wherever the high and low lines crossed there was an interaction of the load and allocation. At that point there was no significant difference between operating at high or low load for that allocation method. Wherever the highest value of the low load line was higher than a value on the high load line there was an interaction. For the Update remote graph it was better to operate at high load with the optimal redundant allocation than under low load with the greatest query allocation.

The slope of the load lines provided information of when the main grouping based on allocation would change based on load. Where the slope of the load line was not continuously negative, the grouping of the allocation methods changed. The Update remote graph clearly showed that for total allocation method the grouping would be best under low load and worst under high load. This was a definite change from the main effect grouping based on just the allocation method's overall average system time.

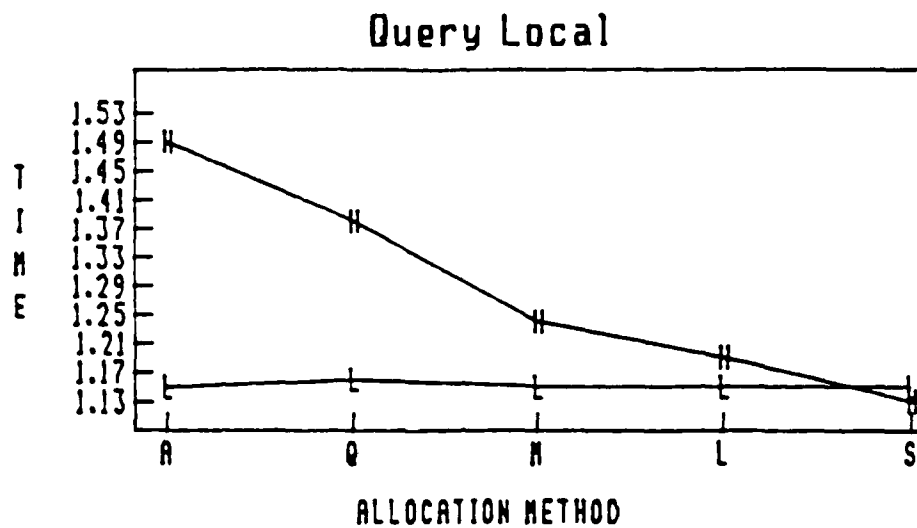


(a) Update Local

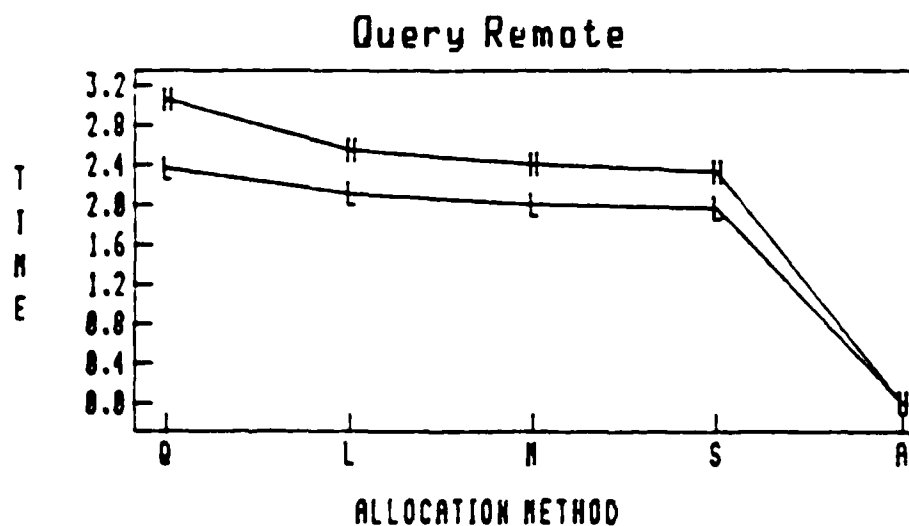


(b) Update Remote

Figure 4.7 Graph of Performance Criterion by Load a) Update Local, b) Update Remote, c) Query Local, d) Query Remote



(c) Query Local



(d) Query Remote

Figure 4.7 Graph of Performance Criterion by Load a) Update Local, b) Update Remote, c) Query Local, d) Query Remote

Decision Function Normalized Data

Figure 4.8a contains the normalized data from the system performance criterion by allocation method and experimental load. As explained in chapter 3, the highest value in the data set (Figure 4.6) was divided into the rest of the set. In the case of high load for Update local the highest value was 1.49. This number was used to divide all of the numbers in that row of Figure 4.6. The results for those calculations were put in the odd rows of column 1 of Figure 4.8a. The entry for all nodes (total replication) at high load is 1.00, since this was the allocation method with the highest value. The same procedure was followed for Update local at low load, with the results put in the even rows of column 1 of Figure 4.8a. Once again the entry for all nodes happened to have had the highest value, so its value in the normalized table was 1.

Figure 4.8b is the normalized data for the storage requirement of each allocation method and the resource cost in terms of the number of operations needed to executed the allocation for 6 nodes and 8 data items. The values to normalize for storage came from the section Storage in this chapter. The highest storage cost was 48 from the total replication method. The normalized value for this allocation method was 1 as shown in Figure 4.8b. The data for normalization of resource costs came from Figure 4.2. The highest value for this cost was 147456 operations from the

optimal redundant allocation method. Its value in the normalized table of Figure 4.8b was 1 as expected. The large magnitude of the number of operations for the optimal redundant case compared to the number of operations for all other allocations, caused the normalized value for all other allocations to be essentially 0.

ALLOCATION*LOAD \	AVERAGE SYSTEM TIME			
	UPLOC	UPREM	QYLOC	QYREM
ALL NODES HIGH	1.00	1.00	1.00	0.00
ALL NODES LOW	1.00	0.69	0.99	0.00
GREATEST QRY HIGH	0.83	0.82	0.92	1.00
GREATEST QRY LOW	0.97	1.00	1.00	1.00
LEAST UPDATE HIGH	0.81	0.78	0.80	0.82
LEAST UPDATE LOW	0.98	0.96	0.99	0.89
OPTIMAL MULT HIGH	0.87	0.68	0.84	0.78
OPTIMAL MULT LOW	0.99	0.77	0.99	0.84
OPTIMAL SNGL HIGH	0.75	0.75	0.76	0.76
OPTIMAL SNGL LOW	0.97	0.94	0.99	0.83

(a) System Performance Criterion

ALLOCATION 2\	STORAGE	RESOURCE
ALL NODES	1.00	0.0
GREATEST QUERY	0.17	0.0
LEAST UPDATE	0.17	0.0
OPTIMAL MULTIPLE	0.35	1.0
OPTIMAL SINGLE	0.17	0.0

(b) Storage and Resource Cost

Figure 4.8. Normalized Data for Decision Function a) System Performance Criterion, b) Storage and Resource Cost Scenarios

The best way to understand the use of the decision function was through some specific cases of values assigned to the weights of the decision criteria in equation 3.1. With the weights from these cases the decision function was executed to

find the relative ranking of the allocation methods for that scenario. The cases examined were a representative group to illustrate the use of the decision function and were by no means an exhaustive set of all possible combinations of weights for this network. The weights for all of the scenarios were summarized in Figure 4.9.

WEIGHTS \	SCENARIO					
	A	B	C	D	E	F
RESPONSE TIME (ti)	0.33	0.70	0.70	1.00	1.00	1.00
HIGH (h)	0.50	0.75	0.75	0.50	0.75	0.75
LOW (l)	0.50	0.25	0.25	0.50	0.25	0.25
UPLOC TIME (t1)	0.25	0.50	0.00	0.25	0.50	0.00
UPREM TIME (t2)	0.25	0.50	0.00	0.25	0.50	0.00
QRYLOC TIME(t3)	0.25	0.00	0.50	0.25	0.00	0.50
QRYREM TIME(t4)	0.25	0.00	0.50	0.25	0.00	0.50
STORAGE (s)	0.33	0.20	0.20	0.00	0.00	0.00
RESOURCE COST (r)	0.33	0.10	0.10	0.00	0.00	0.00

Figure 4.9. Summary of Scenario's Weights

The format for each case was to assign relative weights to the importance of the aggregate response time, storage, and resource cost. Next, relative weights were considered for the load and then weights were figured for the system performance criteria. The results of the decision function for each case are shown in Figure 4.10. The values of Figure 4.10 were given a ranking of 1 (best) for the lowest value and 5 (worst) for the highest value and the rankings were summarized by case in Figure 4.11.

Case A

The network manager considered response time, storage, and resource cost all equally important, therefore a value of .33 was assigned to each of these weights in the decision function. The manager also noted that the network operated just as often at high load as it did at low load, so a value of .5 was assigned to these weights. There were as many queries as update messages that flowed through the network and they were both processed as many times locally as they were remotely, so the value of .25 was assigned to each of these weights. As shown in Figure 4.10 for case A the optimal nonredundant allocation would have been the selection for this network as it had the lowest value of 0.334.

Case B

The network manager considered response time three and one half times more important than storage cost and storage cost was twice as important as resource cost, therefore a value of .70 was assigned to response time weight, a value of .20 was the storage weight, and a value of .10 was assigned the resource weight. The manager also noted that the network operated three times as often at high load as it did at low load, so a value of .75 was assigned to the high load weight and .25 was assigned to the low load weight. There were primarily update messages that flowed through the network and they were both processed as many times locally as they were remotely, so values of .50 were assign to the update weights

and values of .00 were assigned to the query weights. As shown in Figure 4.10 for case B the optimal nonredundant allocation would have been the selection for this network as it had the lowest value of 0.599.

Case C

The network manager considered response time three and one half times more important than storage cost and storage cost was twice as important as resource cost, therefore a value of .70 was assigned to response time weight, a value of .20 was the storage weight, and a value of .10 was assigned the resource weight. The manager also noted that the network operated three times as often at high load as it did at low load, so a value of .75 was assigned to the high load weight and .25 was assigned to the low load weight. There were primarily queries messages that flowed through the network and they were both processed as many times locally as they were remotely, so values of .50 were assign to the query weights and values of .00 were assigned to the update weights. As shown in Figure 4.10 for case C the total redundant allocation would have been the selection for this network as it had the lowest value of 0.550.

Case D

This network was the same as case A, except the manger was not concerned with the cost of storage or the cost of the allocation that would give the best possible system response time possible. Because of the overriding concern for response

time this was assigned a value of 1 and the others were assigned a value of 0. The best allocation method for this scenario was total replication (all nodes) as it had the lowest value of .711.

Case E

This network was the same as case B, except the manger was not concerned with the cost of storage or the cost of the allocation that would give the best possible system response time possible. Because of the overriding concern for response time this was assigned a value of 1 and the others were assigned a value of 0. The best allocation method for this scenario was optimal redundant as it had the lowest value of .802.

Case F

This network was the same as case C, except the manger was not concerned with the cost of storage or the cost of the allocation that would give the best possible system response time possible. Because of the overriding concern for response time this was assigned a value of 1 and the others were assigned a value of 0. The best allocation method for this scenario was total replication (all nodes) as it had the lowest value of .500.

ALLOCATION \	SCENARIO					
	A	B	C	D	E	F
ALL NODES	0.56	0.87	0.55	0.71	0.96	0.50
GREATEST QUERY	0.36	0.64	0.71	0.94	0.87	0.97
LEAST UPDATE	0.34	0.62	0.62	0.88	0.84	0.84
OPTIMAL MULTIPLE	0.72	0.73	0.75	0.84	0.80	0.83
OPTIMAL SINGLE	0.33	0.59	0.59	0.84	0.80	0.79

Figure 4.10. Output of Decision Function by Scenario

ALLOCATION \	SCENARIO					
	A	B	C	D	E	F
ALL NODES	4	5	1	1	5	1
GREATEST QUERY	3	3	4	5	4	5
LEAST UPDATE	2	2	3	4	3	4
OPTIMAL MULTIPLE	5	4	5	3	1	3
OPTIMAL SINGLE	1	1	2	2	2	2

Figure 4.11. Ranking of Allocation Methods by Scenario

Figure 4.11 shows that the optimal multiple (redundant) was only ranked 1 once. This can be explained in part by understanding the term optimal was optimal only in a narrow sense of the transmission time of the network and not the overall delay. It was not knowing the distribution of the network delay that led to the use of the simulation to compare the allocation methods. The large resource cost of the optimal redundant allocation acted as a penalty each time resource was considered as a factor.

The large storage cost of the total redundant allocation also acted as a penalty for this method when storage was considered as in cases A and B. In case C the storage cost penalty was overshadowed by importance of query responses in this network and that for this allocation there is no network cost associated with queries. Whenever queries were of primary concern as in cases C and F or of equal concern with updates and no consideration of storage as in case D, total redundant allocation was the 1st choice.

The ranking of the optimal nonredundant allocation in all cases either 1 or 2 indicates the efficiency of centrally locating the data items in terms of the network median based on the message load per data item and the low cost of storage associated with this allocation method.

V. Conclusions and Recommendations

Conclusions

This thesis analyzed the problem of selecting among different data allocation methods the best one for a distributed database. The selection involved complicated and multiple criteria. There were trade-offs in the different allocation methods criteria, such as faster response times with more storage cost or a more complex allocation method and its associated higher resource cost of computational time.

Data from the Army Maneuver Control system was not readily available for this study. A simulation test network using different data allocation methods was constructed. The simulation had the advantage of allowing a comparison study to be made in the absence of actual data from a distributive database network. Several days of network activity were simulated in just a few hours. The simulation allowed for message delay into the system response times, even though the exact network distribution of delay time was not known.

On the basis of the simulation study and the analysis of the output of the decision function for the given case scenarios, the following conclusions are drawn:

1. The simulation helped gain greater insight into the allocation problem and the factors to be considered in a selection process.

2. The allocation selected changed based on which system performance criteria, such as query remote, was considered.
3. The allocation selected sometimes changed based on the system load.
4. The allocation selected changed based on the importance assigned to the cost of storage and the computational complexity of each allocation method.
5. The decision function proposed included all the factors which affect the selection of one allocation method over another method. The function allowed for weighting of the system response time criteria and system load.
6. The decision function is network independent. Given system data from an actual distributed database network, the decision function could be applied to aid in the complex decision of which data allocation method would be best for the network.
7. The optimal allocation methods were optimal only in the area of network transmission time; not in transmission delay time or storage cost.
8. The pilot work required for each set of simulations was time intensive even for the small test network. The combinations of factors which could be tested for even this size network grew to be complicated quite fast. Only 3 factors required 20 different combinations of simulation runs.

Recommendations

Based on the assumptions in chapter 1 and the lessons learned during this study , the following recommendations are made:

1. The automated methods of data allocation need to be expanded to include the storage constraint for each node.
2. The automated methods of data allocation need to be expanded to include the constraint of the user or in the case of MCS the Commander designating certain critical data items to be assigned to particular nodes.

3. The automated methods of data allocation need to be adapted to allow for storing data based on a priority scheme of those items deemed critical by the user or Commander first and then allocating noncritical data items.
4. The automated methods need to have a routine added which would measure execution time to within 1/100th of a second. This time could be used as a basis for cost of resource (in terms of time) to execute an allocation method.
5. The simulation model needs to be enlarged to study the impact of different network configurations on the system performance criteria and the different allocation methods.
6. The simulation model needs to be enlarged to study the impact of the network configurations being unreliable on the system performance criteria and the different allocation methods.
7. MCS estimates of update and query rates and system processing times need to be made since they were not available for this study or for validation of a simulation model specifically for the MCS.
8. The optimal nonredundant method assigned a data item based on the median cost node of these two message types. A possible allocation heuristic based on the ratio of update to query messages at a node or collection of nodes should be explored.
9. A statistical package, such as SAS on the CSC at AFIT, should be installed on the ICC.
10. The Fortran 77 compiler on the ICC should be upgraded to the same version as on the SSC or ASC.

Summary

The allocation of data in a distributed database is a complex problem. Many issues such as replication of data to ensure survivability of a military network were not covered in this research. This thesis effort addressed multiple factors impacting on an allocation scheme such as message response time, storage cost, and resource cost to execute the allocation

method. A framework or methodology for relating and evaluating the multiple and sometimes conflicting factors was presented in the form of a decision function. The decision function can be used in future research in evaluating new proposed heuristics in data allocation or used in evaluating the impact of any additional factors addressed in the recommendations. The decision function can be used now by network managers as a decision tool in selecting among competing data allocation schemes for an existing distributed system.

Appendix A: Source Code

For the source code for the automated allocation methods
and the SLAM II network code for the 6 node network
configuration write to:

Air Force Institute of Technology
School of Engineering
ATTN: Dr. Hartrum AFIT/ENG
Wright Patterson AFB, Ohio 45433.

Appendix B: ANOVA Tables for Simulation Data

SAS, a statistical package, was used to perform an analysis of variance (ANOVA) which tested the null hypothesis that there was no difference for any of the system performance criterion due to system load, allocation method, queuing discipline, or their interactions. SAS was also used to run Duncan multiple-range tests on all main effects of the model with the queuing effect removed. All tests and analysis was performed at a 95% confidence level.

Figures B.1 - B.4 are the ANOVA tables of the null hypothesis for system performance criterion of messages update local, update remote, query local, and query remote also referred by the labels T1, T2, T3, and T4 respectively. The $P > F$ column for the queuing factor and all interactions with queuing is greater than 0.05. This means these factors were not significant to the model for any of the performance criterion times.

The SAS program model was modified to include the queuing factor and its interaction in with the sum of error to make a more simple model. Figures B.5 - B.8 are the ANOVA tables of the modified model of significant factors only. The R-Square value for the new model for each system time was close to the old model verifying that the simpler model still explained the data variance well. The R-Square of the model was expected to

be higher for T2 and T4 as these times included network transmission time as addressed by all the allocation methods.

DEPENDENT VARIABLE: T1		UPDATE LOCAL		
CLASS	LEVELS	VALUES		
ALLOC	5	ALL NODES,GREATEST QUERY,LEAST UPDATE, OPTIMAL MULTIPLE,OPTIMAL SINGLE		
LOAD	2	HIGH,LOW		
QUEUE	2	FIFO,PRIORITY		
NUMBER OF OBSERVATIONS IN DATA SET = 120				
SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	
MODEL	19	1.34063119	0.07055954	
ERROR	100	0.20511883	0.00205119	
CORRECTED TOTAL	119	1.54575002		
MODEL F = 34.40		PR > F = 0.0001		
R-SQUARE	C.V.	ROOT MSE	T1 MEAN	
0.867301	3.7336	0.04529005	1.21305576	
SOURCE	DF	ANOVA SS	F VALUE	PR > F
ALLOC	4	0.53998610	65.81	0.0001
LOAD	1	0.42044112	204.97	0.0001
ALLOC*LOAD	4	0.37374611	45.55	0.0001
QUEUE	1	0.00015386	0.08	0.7847
ALLOC*QUEUE	4	0.00412646	0.50	0.7336
LOAD*QUEUE	1	0.00089589	0.44	0.5102
ALLOC*LOAD*QUEUE	4	0.00128165	0.16	0.9598

Figure B.1. ANOVA Update Local for Null Hypothesis

DEPENDENT VARIABLE: T2 UPDATE REMOTE

CLASS	LEVELS	VALUES
ALLOC	5	ALL NODES,GREATEST QUERY,LEAST UPDATE, OPTIMAL MULTIPLE,OPTIMAL SINGLE
LOAD	2	HIGH,LOW
QUEUE	2	FIFO,PRIORITY

NUMBER OF OBSERVATIONS IN DATA SET = 120

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE
MODEL	19	9.33018578	0.49106241
ERROR	100	0.48252226	0.00482522
CORRECTED TOTAL	119	9.81270804	

MODEL F = 101.77

PR > F = 0.0

R-SQUARE	C.V.	ROOT MSE	T2 MEAN
0.950827	4.3428	0.06946382	1.59952437

SOURCE	DF	ANOVA SS	F VALUE	PR > F
ALLOC	4	1.69922999	88.04	0.0001
LOAD	1	3.84018260	795.86	0.0
ALLOC*LOAD	4	3.78715002	196.22	0.0
QUEUE	1	0.00032560	0.07	0.7956
ALLOC*QUEUE	4	0.00205646	0.11	0.9800
LOAD*QUEUE	1	0.00015738	0.03	0.8570
ALLOC*LOAD*QUEUE	4	0.00108372	0.06	0.9940

Figure B.2. ANOVA Update Remote for Null Hypothesis

DEPENDENT VARIABLE: T3

QUERY LOCAL

CLASS	LEVELS	VALUES
ALLOC	5	ALL NODES,GREATEST QUERY,LEAST UPDATE, OPTIMAL MULTIPLE,OPTIMAL SINGLE
LOAD	2	HIGH,LOW
QUEUE	2	FIFO,PRIORITY

NUMBER OF OBSERVATIONS IN DATA SET = 120

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE
MODEL	19	1.54960446	0.08155813
ERROR	100	0.29477322	0.00294773
CORRECTED TOTAL	119	1.84437768	

MODEL F = 27.67 PR > F = 0.0001

R-SQUARE	C.V.	ROOT MSE	T3 MEAN
0.840177	4.4579	0.05429302	1.21790591

SOURCE	DF	ANOVA SS	F VALUE	PR > F
ALLOC	4	0.53332573	45.23	0.0001
LOAD	1	0.54477721	184.81	0.0001
ALLOC*LOAD	4	0.46818605	39.71	0.0001
QUEUE	1	0.00112576	0.38	0.5380
ALLOC*QUEUE	4	0.00091147	0.08	0.9890
LOAD*QUEUE	1	0.00028570	0.10	0.7562
ALLOC*LOAD*QUEUE	4	0.00099254	0.08	0.9871

Figure B.3. ANOVA Query Local for Null Hypothesis

DEPENDENT VARIABLE: T4

QUERY REMOTE

CLASS	LEVELS	VALUES
ALLOC	5	ALL NODES, GREATEST QUERY, LEAST UPDATE, OPTIMAL MULTIPLE, OPTIMAL SINGLE
LOAD	2	HIGH, LOW
QUEUE	2	FIFO, PRIORITY

NUMBER OF OBSERVATIONS IN DATA SET = 120

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE
MODEL	19	116.15542910	6.11344364
ERROR	100	0.95117374	0.00951174
CORRECTED TOTAL	119	117.10660283	

MODEL F = 642.73

PR > F = 0.0

R-SQUARE	C.V.	ROOT MSE	T4 MEAN
0.991878	5.2008	0.09752814	1.87524922

SOURCE	DF	ANOVA SS	F VALUE	PR > F
ALLOC	4	110.22873311	2897.18	0.0
LOAD	1	4.37256060	459.70	0.0001
ALLOC*LOAD	4	1.53899829	40.45	0.0001
QUEUE	1	0.00869100	0.91	0.3414
ALLOC*QUEUE	4	0.00268773	0.07	0.9908
LOAD*QUEUE	1	0.00054297	0.06	0.8117
ALLOC*LOAD*QUEUE	4	0.00321540	0.08	0.9870

Figure B.4. ANOVA Query Remote for Null Hypothesis

DEPENDENT VARIABLE: T1		UPDATE LOCAL		
CLASS	LEVELS	VALUES		
ALLOC	5	ALL NODES,GREATEST QUERY,LEAST UPDATE, OPTIMAL MULTIPLE,OPTIMAL SINGLE		
LOAD	2	HIGH,LOW		
QUEUE	2	FIFO,PRIORITY		
NUMBER OF OBSERVATIONS IN DATA SET = 120				
SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	
MODEL	9	1.33417333	0.14824148	
ERROR	110	0.21157669	0.00192342	
CORRECTED TOTAL	119	1.54575002		
MODEL F =	77.07	PR > F = 0.0		
R-SQUARE	C.V.	ROOT MSE	T1 MEAN	
0.863124	3.6154	0.04385686	1.21305576	
SOURCE	DF	ANOVA SS	F VALUE	PR > F
ALLOC	4	0.53998610	70.19	0.0001
LOAD	1	0.42044112	218.59	0.0001
ALLOC*LOAD	4	0.37374611	48.58	0.0001

Figure B.5. ANOVA Update Local for Significant Factors

DEPENDENT VARIABLE: T2		UPDATE REMOTE		
CLASS	LEVELS	VALUES		
ALLOC	5	ALL NODES,GREATEST QUERY,LEAST UPDATE, OPTIMAL MULTIPLE,OPTIMAL SINGLE		
LOAD	2	HIGH,LOW		
QUEUE	2	FIFO,PRIORITY		
NUMBER OF OBSERVATIONS IN DATA SET = 120				
SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	
MODEL	9	9.32656262	1.03628474	
ERROR	110	0.48614542	0.00441950	
CORRECTED TOTAL	119	9.81270804		
MODEL F = 234.48		PR > F = 0.0		
R-SQUARE	C.V.	ROOT MSE	T2 MEAN	
0.950458	4.1562	0.06647935	1.59952437	
SOURCE	DF	ANOVA SS	F VALUE	PR > F
ALLOC	4	1.69922999	96.12	0.0001
LOAD	1	3.84018260	868.92	0.0
ALLOC*LOAD	4	3.78715002	214.23	0.0

Figure B.6. ANOVA Update Remote for Significant Factors

DEPENDENT VARIABLE: T3		QUERY LOCAL		
CLASS	LEVELS	VALUES		
ALLOC	5	ALL NODES,GREATEST QUERY,LEAST UPDATE, OPTIMAL MULTIPLE,OPTIMAL SINGLE		
LOAD	2	HIGH,LOW		
QUEUE	2	FIFO,PRIORITY		
NUMBER OF OBSERVATIONS IN DATA SET = 120				
SOURCE	DF	SUM OF SQUARES	MEAN SQUARE	
MODEL	9	1.54628899	0.17180989	
ERROR	110	0.29808869	0.00270990	
CORRECTED TOTAL	119	1.84437768		
MODEL F = 63.40		PR > F = 0.0		
R-SQUARE	C.V.	ROOT MSE	T3 MEAN	
0.838380	4.2743	0.05205667	1.21790591	
SOURCE	DF	ANOVA SS	F VALUE	PR > F
ALLOC	4	0.53332573	49.20	0.0001
LOAD	1	0.54477721	201.03	0.0001
ALLOC*LOAD	4	0.46818605	43.19	0.0001

Figure B.7. ANOVA Query Local for Significant Factors

DEPENDENT VARIABLE: T4

QUERY REMOTE

CLASS	LEVELS	VALUES
ALLOC	5	ALL NODES,GREATEST QUERY,LEAST UPDATE, OPTIMAL MULTIPLE,OPTIMAL SINGLE
LOAD	2	HIGH,LOW
QUEUE	2	FIFO,PRIORITY

NUMBER OF OBSERVATIONS IN DATA SET = 120

SOURCE	DF	SUM OF SQUARES	MEAN SQUARE
MODEL	9	116.14029200	12.90447689
ERROR	110	0.96631084	0.00878464
CORRECTED TOTAL	119	117.10660283	

MODEL F = 1468.98 PR > F = 0.0

R-SQUARE	C.V.	ROOT MSE	T4 MEAN
0.991748	4.9981	0.09372643	1.87524922

SOURCE	DF	ANOVA SS	F VALUE	PR > F
ALLOC	4	110.22873311	3136.97	0.0
LOAD	1	4.37256060	497.75	0.0
ALLOC*LOAD	4	1.53899829	43.80	0.0001

Figure B.8. ANOVA Query Remote for Significant Factors

Figure B.5 through B.8 all showed in the column "PR > F" for the load and allocation factor and their interaction, values less than 0.050. This meant the factors were significant at the 5% alpha level. Based on this fact, the model was modified to provide a post hoc test to tell which factors were significantly different from each other. The Duncan multiple-range test was a post hoc test available in SAS.

Letters are used in the Duncan multiple-range test output to show factor groupings. Factors with the same letter are not significantly different from each other. The Duncan multiple-range test compares differences of means by groups. The means of the factors are part of the Duncan grouping output. Figure B.9, B.11, B.13 and B.15 are summary output of the SAS Duncan grouping for Update local, Update remote, Query local, and Query remote respectively. Figures B.10, B.12, B.14, and B.16 are summary output of the Duncan grouping computed outside of SAS for the interaction effects between allocation and load.

<u>DUNCAN GROUPING</u>	<u>MEAN T1</u>	<u>N</u>	<u>ALLOCATION</u>
A	1.33245	24	ALL NODES
B	1.22931	24	OPTIMAL MULTIPLE
C	1.19354	24	GREATEST QUERY
C	1.17565	24	LEAST UPDATE
D	1.13432	24	OPTIMAL SINGLE

<u>DUNCAN GROUPING</u>	<u>MEAN T1</u>	<u>N</u>	<u>LOAD</u>
A	1.272248	60	HIGH
B	1.153864	60	LOW

Figure B.9. Duncan Range Test for Update Local Main Effects

<u>ALLOCATION</u>	<u>LOAD</u>	<u>N</u>	<u>MEAN</u>	<u>T1</u>	<u>DUNCAN GROUPING</u>
ALL NODES	HIGH	12	1.49141		A
OPTIMAL MULTIPLE	HIGH	12	1.30053		B
GREATEST QUERY	HIGH	12	1.23932		C
LEAST UPDATE	HIGH	12	1.20067		D
ALL NODES	LOW	12	1.17348		D E
OPTIMAL MULTIPLE	LOW	12	1.15807		E F
LEAST UPDATE	LOW	12	1.15063		E F
GREATEST QUERY	LOW	12	1.14775		E F
OPTIMAL SINGLE	LOW	12	1.13935		E F
OPTIMAL SINGLE	HIGH	12	1.12927		F

Figure B.10. Duncan Range Test for interaction Update Local

<u>DUNCAN GROUPING</u>	<u>MEAN T2</u>	<u>N</u>	<u>ALLOCATION</u>
A	1.72225	24	GREATEST QUERY
B	1.65971	24	ALL NODES
B	1.64354	24	LEAST UPDATE
C	1.59659	24	OPTIMAL SINGLE
D	1.37554	24	OPTIMAL MULTIPLE

<u>DUNCAN GROUPING</u>	<u>MEAN T2</u>	<u>N</u>	<u>LOAD</u>
A	1.77841	60	HIGH
B	1.42063	60	LOW

Figure B.11. Duncan Range Test for Update Remote Main Effects

<u>ALLOCATION</u>	<u>LOAD</u>	<u>N</u>	<u>MEAN</u>	<u>T2</u>	<u>DUNCAN GROUPING</u>
ALL NODES	HIGH	12	2.19256		A
GREATEST QUERY	HIGH	12	1.81773		B
LEAST UPDATE	HIGH	12	1.72815		C
OPTIMAL SINGLE	HIGH	12	1.66368		D
GREATEST QUERY	LOW	12	1.62677		E
OPTIMAL SINGLE	LOW	12	1.52949		F
LEAST UPDATE	LOW	12	1.55891		F
OPTIMAL MULTIPLE	HIGH	12	1.48993		G
OPTIMAL MULTIPLE	LOW	12	1.26114		H
ALL NODES	LOW	12	1.12684		I

Figure B.12. Duncan Range Test for Interaction Update Remote

<u>DUNCAN GROUPING</u>	<u>MEAN T3</u>	<u>N</u>	<u>ALLOCATION</u>
A	1.32025	24	ALL NODES
B	1.26789	24	GREATEST QUERY
C	1.19427	24	OPTIMAL MULTIPLE
D C	1.16787	24	LEAST UPDATE
D	1.13925	24	OPTIMAL SINGLE
<u>DUNCAN GROUPING</u>	<u>MEAN T3</u>	<u>N</u>	<u>LOAD</u>
A	1.285284	60	HIGH
B	1.150528	60	LOW

Figure B.13. Duncan Range Test for Query Local Main Effects

<u>ALLOCATION</u>	<u>LOAD</u>	<u>N</u>	<u>MEAN</u>	<u>T3</u>	<u>DUNCAN GROUPING</u>
ALL NODES	HIGH	12	1.48590		A
GREATEST QUERY	HIGH	12	1.37786		B
OPTIMAL MULTIPLE	HIGH	12	1.24079		C
LEAST UPDATE	HIGH	12	1.18937		D
GREATEST QUERY	LOW	12	1.15791		D E
ALL NODES	LOW	12	1.15459		E
OPTIMAL MULTIPLE	LOW	12	1.14775		E
LEAST UPDATE	LOW	12	1.14637		E
OPTIMAL SINGLE	LOW	12	1.14600		E
OPTIMAL SINGLE	HIGH	12	1.13248		E

Figure B.14. Duncan Range Test for interaction Query Local

DUNCAN GROUPING	MEAN T4	N	ALLOCATION
A	2.71020	24	GREATEST QUERY
B	2.12967	24	LEAST UPDATE
C	2.19092	24	OPTIMAL MULTIPLE
C	2.14545	24	OPTIMAL SINGLE
D	0.00000	24	ALL NODES
DUNCAN GROUPING	MEAN T4	N	LOAD
A	2.06614	60	HIGH
B	1.68436	60	LOW

Figure B.15. Duncan Range Test for Query Remote Main Effects

ALLOCATION	LOAD	N	MEAN	T4	DUNCAN GROUPING
GREATEST QUERY	HIGH	12	3.06594		A
LEAST UPDATE	HIGH	12	2.54276		B
OPTIMAL MULTIPLE	HIGH	12	2.38588		C
GREATEST QUERY	LOW	12	2.35445		C D
OPTIMAL SINGLE	HIGH	12	2.33608		D
LEAST UPDATE	LOW	12	2.11658		E
OPTIMAL MULTIPLE	LOW	12	1.99595		F
OPTIMAL SINGLE	LOW	12	1.95481		G
ALL NODES	HIGH	12	0.00000		H
ALL NODES	LOW	12	0.00000		H

Figure B.16. Duncan Range Test for interaction Query Remote

Bibliography

- Athans, Michael and Mooses Ma. "Optimal File Allocation Problems for Distributed Data Bases in Unreliable Computer Networks II," Report AD-A138, Cambridge: Massachusetts Institute of Technology, December 1983.
- Chu, Wesley W. "Optimal File Allocation in a Multiple Computer System," IEEE Transactions on Computers, 18:885-889, (1969).
- Cody, Ronald P. and Jeffrey K. Smith. Applied Statistics and the SAS Programing Language. New York: North-Holland, 1985.
- Harwood, Odus E. "Analysis and Development of a Distributed Data Base Design Methodology for the United States Army Maneuver Control System". AFIT thesis AFIT/GCS ENG/86J-3, 1986.
- Korth, Henry and Abraham Silberschatz. Database System Concepts. New York: McGraw Hill Book Company, 1986.
- Levin, K. Dan and Howard L. Morgan. "Optimal Program and Data Locations in Computer Networks," Communications of the ACM, 5:432-439, Association for Computing Machinery, New York, May 1977.
- Myers, Raymond H. and Ronald E. Walpole. Probability and Statistics for Engineers and Scientists. New York: Macmillian, 1972.
- Pritsker, A. Alan B. Introduction to Simulation and SLAM II (Third Edition). New York: John Wiley & Sons, 1986.
- Ramamoorthy, C. V., et al. "Architectural Issues in Distributed Data Base Systems," Proceedings of the Third International Conference on Very Large Data Bases, IEEE Computer Society: 121-126 (October 1977).
- Tannenbaum, Andrew S. Computer Networks. Englewood Cliffs: Prentice-Hall, 1981.
- Ullman, Jeffery D. Principles of Database Systems (Second Edition). Rockville: Computer Science Press, 1982.

VITA

Captain Edward T. Poore Jr. was born on 9 February 1956 in Seattle, Washington. He graduated from Decatur High School in 1974 and attended the United States Military Academy. Upon graduation from the United States Military Academy in June 1978, he earned the degree of Bachelor of Science and was commissioned as a Second Lieutenant in the United States Army Signal Corps. His previous assignments in the United States Army were Telecommunications Platoon Leader, S-4 and Teleprocessing Operations Officer, 50th Signal Battalion Airborne, Fort Bragg, North Carolina; S-2/S-3, Communications-Electronics Department, Combined Arms Training Center, Bad Toelz, West Germany; and Commander of Company A, 7th Army Training Command, Bad Toelz, West Germany. Captain Poore's military schooling included the Communications-Electronics Staff Officer Course, Signal Officer Advance Course, and the Teleprocessing Operations Officer Course. He entered the school of Engineering, Air Force Institute of Technology, in June 1986.

Permanent address: 3824 S. 348th

Auburn, Washington 98001

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT SECURITY CLASSIFICATION CLASSIFIED			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GCS/ENG/87D-21			5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION School of Engineering		6b. OFFICE SYMBOL (If applicable) AFIT/ENG		7a. NAME OF MONITORING ORGANIZATION	
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright Patterson AFB OH 45433-6583			7b. ADDRESS (City, State, and ZIP Code)		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Project Manager Opns Tact Data System		8b. OFFICE SYMBOL (If applicable) ANCPM-OTDS		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code) Fort Monmouth, New Jersey 07703			10. SOURCE OF FUNDING NUMBERS		
			PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.
			WORK UNIT ACCESSION NO.		
11. TITLE (Include Security Classification) See Box 19					
12. PERSONAL AUTHOR(S) Edward T. Poore Jr, B.S., Captain, USA					
13a. TYPE OF REPORT MS Thesis		13b. TIME COVERED FROM _____ TO _____		14. DATE OF REPORT (Year, Month, Day) 1987 December	
15. PAGE COUNT 93					
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	Tactical Data System, Communications Network Distributed Databases, Simulation, Information Systems, Command and Control, <i>theses.</i>		
05	01				
09	02				
19. ABSTRACT (Continue on reverse if necessary and identify by block number)					
<p>Title: DESIGN METHODOLOGY FOR ALLOCATING DATA IN A DISTRIBUTIVE DATABASE (UNCLASSIFIED)</p> <p>Thesis Chairman: Thomas C. Hartrum, PhD Associate Professor of Electrical and Computer Engineering</p> <p style="text-align: right;"><i>John Wilson</i> 31 Dec 87 Development Wright-Patterson AFB OH 45433</p>					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL Thomas C. Hartrum			22b. TELEPHONE (Include Area Code) (513) 255-3576		22c. OFFICE SYMBOL AFIT/ENG

UNCLASSIFIED

BLOCK 19. ABSTRACT

The objective of this research was to find a method to solve the problem of deciding the best means for allocating data in a distributed data base. A decision function which considered system response times of update and query messages, storage cost of data items, and execution cost for allocating data was proposed. Additionally, the decision function allowed weighting of decision criteria to tailor the selection to specific network conditions. Five data allocation methods were evaluated for a given test network. The allocation methods were automated to facilitate future research efforts.

The test network was implemented as a simulation model consisting of six nodes and eight data items. The simulation model was both verified and validated. Statistical analysis was performed on selected outputs of the model. Normalized data from the simulation model, normalized output of the automated allocation methods and scenarios of weighted decision criteria for certain network conditions were evaluated by the decision function. The utility of the decision function was demonstrated through comparison of the results of each scenario. The impact of the research results were discussed and areas of future research were presented.

UNCLASSIFIED

END

3-88

DTIC